
# Multilingual Processing of Temporal Expressions: Detection and Normalisation

**Author:** Nayla Belén García Escribano

**Advisors:** Rodrigo Agerri and German Rigau

hap/lap

Hizkuntzaren Azterketa eta Prozesamendua
Language Analysis and Processing

## Final Thesis

February 2021

---

**Departments**: Computer Systems and Languages, Computational Architectures and Technologies, Computational Science and Artificial Intelligence, Basque Language and Communication, Communications Engineer.

---

**Abstract**

Temporal information processing allows us to identify when an event happened, to temporally relate different events and to create timelines with all the extracted information. To achieve these goals, detecting time expressions in a text (e.g. *the week of March 6*) and normalising them to a standardised value (e.g. a concrete week of a concrete year) is a fundamental step. In this work, we have developed a modular system for time expression processing in English and Spanish, which consists of a Flair sequence-labelling detector based on neural networks and a normaliser built on the TimeNorm synchronous context-free grammar system. Furthermore, we provide an exhaustive study on how we built our Spanish TimeNorm grammar, which is our main contribution and could be useful to adapt the normalisation system to other inflected languages. We evaluated this approach on the TempEval-3 challenge and obtained state-of-the-art results both in detection and normalisation for English and Spanish.

**Keywords:** Temporal Normalization, Temporal Detection, Information Extraction, Natural Language Processing

# Contents

# List of Figures

# List of Tables

# 1 Introduction

Many Natural Language Processing (NLP) tasks involve the extraction of different kinds of information from large amounts of texts. In particular, it may be useful to know when an event happened, to relate several events regarding their times or to build timelines with all the temporal information that we can get from a text. Temporal information processing allows us to do all these things automatically. Some of the many NLP applications are: question-answering, by providing information about when something happened in absolute time or in relation to which other event (Saquete et al., 2009); text summarisation, identifying and ordering different events to generate a representation of a text (Daniel and Allison, 2004); or information extraction, for example, finding relevant temporally-anchored events in patient records when looking for causes of disease (Lin et al., 2015).

An essential step in temporal information processing is the extraction of time expressions. A time expression, also called timex (timexes in plural), is a linguistic sequence that refers to a time, a date, a duration or a frequency, according to the markup language TimeML (Pustejovsky et al., 2005). For example, *6 de marzo* (*6th March*) in the sentence below is a timex that denotes a date and could indicate the 6th day of March of the year 2021, but also of any other year. In this case, we could think that it points at a year close to the moment when the sentence was uttered.

1. La reunión será el *6 de marzo*, *un mes después* de la anterior.
   (The meeting will be the *6th March*, *a month after* the last one.)

Processing time expressions like this one helps us disambiguate the specific time it refers to (e.g. the exact year), temporally contextualising the related events (e.g. a meeting that is supposed to be that date) and organising many events on a timeline (e.g. this day meeting is supposed to be one month after the last one). Thus, a comon purpose of timex extraction is thus to reduce ambiguities and to locate in time some events with respect to others. However, dealing with time expressions is not a straightforward task.

The TempEval-3 challenge (UzZaman et al., 2012), from the SemEval series, proposes a task to evaluate English and Spanish timex extraction. This task consists of two subtasks: first, a detection phase to find the spans of time expressions in raw texts (e.g. *6 de marzo*) and classify them by type (e.g. DATE); and, second, a normalisation phase to associate the detected timexes to the corresponding standardised values (e.g. "2021-03-06").

In this project we have developed a system that can perform this detection and normalisation task both in English and Spanish. To that end, we trained a timex detector for each language using the Flair sequence-labelling framework (Akbik et al., 2018) and we adapted the TimeNorm synchronous context-free grammar (SCFG) normaliser (Bethard, 2013b) to Spanish (English was already implemented). This last part, which consisted in building a Spanish grammar for timex normalisation, is the most important contribution of this thesis and is therefore extensively explained. Unlike other approaches to temporal normalisation (Strötgen et al., 2013), TimeNorm offers a modular approach that does not include the detection component. Thus, it allows us to choose the most appropriate approach or technique for timex detection.

Our work has been performed under the hypothesis that such a grammar can be constructed on a framework designed for English and can achieve good performance at timex normalisation. The objective of the project was to build a robust Spanish grammar that can obtain state-of-the-art results not only at normalisation but also in combination with timex detection. Moreover, it is meant to be a model for the construction of grammars in other inflected languages, along with the Italian proposal of Mirza and Minard (2015). We also aimed at the creation of a timex detector for each language that could allow us to evaluate the whole system on TempEval-3 with comparable results to the participants.

On completion of the project, we summarise our contributions:

- a detailed methodology to build timex normalisation grammars for Spanish that achieves state-of-the-art results in the evaluation on TimeBank,
- a neural timex detector that obtains state-of-the-art results in English and Spanish and that can be combined with the TimeNorm normalisers,
- an exhaustive study on the most relevant difficulties and complex cases that we found during the construction of our Spanish grammar on TimeNorm, and
- every resource and model used for temporal detection and normalisation will be made publicly available.

The document structure is the following. In Section 2, we contextualise temporal information processing with special regard to TimeML (Pustejovsky et al., 2005), the TempEval-3 challenge (UzZaman et al., 2012) and the TimeNorm system (Bethard, 2013b). Section 3 describes our system and the resources used, as well as the evaluation methods. Then, in Section 4, we explain how the TimeNorm SCFG works and how we dealt with all the casuistry of the Spanish grammar construction. Finally, Section 5 reports the empirical results obtained on both timex detection and normalisation.

# 2 Related Work

In this section, we will get a brief glance at the major achievements in temporal information processing with particular regard to time expression extraction. A summary of the main milestones is provided right below. Then, we will comment on some of the most important annotation standards and their respective corpora. Finally, we will end by explaining shortly the TempEval-3 Challenge and the TimeNorm project, as they are significant to this work.

Temporal information processing has been a goal in information extraction since the late 1990's. As Setzer and Gaizauskas (2000) summarise, among the first attempts we find the Message Understanding Conferences (MUC), which proposed named entity subtasks for detecting absolute time expressions (Kaufman, 1995) and relative time expressions (Kaufman, 1998) in text, as well as an unsuccessful subtask for assigning dates to events (Kaufman, 1998). The work of Setzer and Gaizauskas (2000) itself, along with Ferro et al. (2001)'s, provided solid annotation schemes that would after inspire the TimeML specification language (Pustejovsky et al., 2005).

After the first approaches based on MUC, the TempEval tasks in SemEval became the reference framework for temporal information processing, and the TimeML-based TimeBank (Pustejovsky et al., 2003), the reference corpus. The TempEval challenges have focused on newswire texts (Verhagen et al., 2009, 2010; UzZaman et al., 2012) and more recently on clinical texts (Bethard et al., 2015, 2016), but always maintaining the TimeML schema as basis. An alternative and deeper annotation language has been presented by Bethard and Parker (2016), along with new detection and normalisation systems (Laparra et al., 2018; Xu et al., 2019).

## 2.1 Temporal Annotation

Temporal annotation consists in annotating temporal information, either in the form of expressions from the text or relations among different text sequences. The main objective is to provide a formal representation of the temporal content in a text, which will allow us, for example, to find relations between the referred events or generate timelines with all the extracted information.

### 2.1.1 Annotation Standards

One of the first attempts to create an annotation language for temporal information is the proposal from Mani and Wilson (2000). This scheme was designed to represent time expressions regarding their dependency on the speaker and the reference time, which entailed a progress with respect to MUC subtasks. However, their approach was limited to timex normalisation referring to dates and times.

The work of Setzer and Gaizauskas (2000) sets part of the bases for TimeML, for example, introducing a specification for events and temporal relations. Their approach is more focused on events, as they aim at event ordering. On the other hand, the TIDES

schema (Ferro et al., 2001) defines time expressions as TIMEX2 elements, which present a "value" attribute to assign normalisation values as ISO representations in the ISO 8601 standard. These "value" attributes and ISO representations will be inherited by TimeML TIMEX3 elements.

The TimeML specification language (Pustejovsky et al., 2005) was developed to represent the content of time expressions and events based on the advances in named entity recognition (NER) and the lack of a markup system for real content extraction. The design improves the TIMEX2 (Ferro et al., 2001) and Setzer and Gaizauskas (2000) annotation systems. Unlike other schemes, TimeML separates the sequences that convey time or event information from their anchoring or other ordering dependencies. From the different data structures that TimeML specifies, TIMEX3 is the one that is used to annotate time expressions such as times, dates, durations and sets (frequencies), inspired by TIMEX2. TIMEX3 elements can be identified with regard to an anchor and can be related to events or other time expressions. More on timexes and TIMEX3 elements will be explained in Section 3.2, as the TempEval-3 challenge, the TimeNorm system and our work are based on TimeML.

Within the TimeNorm project, Bethard and Parker (2016) propose a new semantically compositional annotation scheme to cover the weaknesses of TimeML on normalisation. Mainly, it makes possible to represent times: a) that do not align to a single calendar unit (day, week, month, etc.), like *the last three summers*; b) that are relative to events and not only to other times, such as *three weeks postoperative*; and c) that are built compositionally by operating with smaller times, like the union composed by *Tuesdays and Thursdays*. This approach aims at a wider and more fine-grained interpretation of time expressions by translating their semantics into mathematical operations over intervals on a timeline. The representations from the first TimeNorm system (Bethard, 2013b), which will be explained more deeply in Sections 3 and 4, present the early versions of some of these compositional rules.

### 2.1.2 Corpora

The TimeBank corpus (Pustejovsky et al., 2003), which was built along with the TimeML scheme, is a gold-standard human-annotated corpus designed for temporal expression, event and relation annotation based on TimeML. All its corpora files are newswire texts. The TimeML markup language has also been used to create other corpora like TIMEN (Llorens et al., 2012), AQUAINT (Graff, 2002) or the first three TempEval challenges (Verhagen et al., 2009, 2010; UzZaman et al., 2012). The Clinical TempEval tasks (Bethard et al., 2015, 2016) were evaluated on the THYME clinical corpus (Styler et al., 2014), that is based on an extended version of TimeML. On the other hand, Bethard and Parker (2016) re-annotated TimeBank with their scheme to build the Semantically Compositional Annotation of Time Expressions (SCATE) corpus.

As it can be expected, all the referred annotation schemes have been designed for the English language and the mentioned corpora gather only English texts. Nevertheless, TimeML has been the base for textual resources in other languages, like the Spanish

TimeBank developed by Saurí and Badia (2012) and used in TempEval-2 (Verhagen et al., 2010) and TempEval-3 (UzZaman et al., 2012), or Ita-TimeBank (Caselli et al., 2011), a TimeML corpus for Italian that was used by Mirza and Minard (2015).

## 2.2 Temporal Information Processing

Temporal information processing covers several tasks that are mainly focused on time expressions, event anchoring, temporal relations among timexes and events and generation of timelines. Regarding time expressions alone, tasks usually involve detection and/or normalisation, as our system does.

### 2.2.1 The TempEval Challenge

The TempEval challenges, as we have seen, have been the reference for temporal information processing along with the TimeML annotation basis. The first TempEval (Verhagen et al., 2009) evaluated temporal relations directly, but it resulted in very difficult tasks. For this reason, TempEval-2 (Verhagen et al., 2010) and TempEval-3 (UzZaman et al., 2012) included timex detection and normalisation, as well as event detection and classification. These two evaluated in English and Spanish, although TempEval-2 prepared the challenge for other four languages that did not get any participants. Later on, the two Clinical TempEval challenges (Bethard et al., 2015, 2016) prepared very similar evaluations, in which timex normalisation was not considered. Thus, the last TempEval reference for normalisation is still TempEval-3. Apart from all these tasks, other usual exercises since TempEval-2 involved the detection and classification of events and identifying temporal relations with different variations.

Among the best systems in TempEval-3 for timex detection and normalisation, HeidelTime (Strötgen and Gertz, 2010; Strötgen et al., 2013) is the most prominent. This is a rule-engineered system that has been designed not only for timex extraction, but also for the exploration of spatio-temporal information. The HeidelTime system won the TempEval-2 detection and normalisation tasks, whereas in TempEval-3 got the best scores on timex normalisation for both Spanish and English and was the best on Spanish timex detection. A more recent version of HeidelTime (Strötgen and Gertz, 2015) presents a way to automatically extend the system for any language.

Other notable rule-based systems are NavyTime (Chambers, 2013) and SUTime (Chang and Manning, 2012), which got good results in English timex detection. However, all participants used rule-based methods for normalisation, including those which are data-driven for detection, like ClearTK (Bethard, 2013a) or TIPSem (Llorens et al., 2010). This last system is one of the few that is also available for Spanish, and it served as reference in TempEval-3 for both languages. According to the results in TempEval-2 and TempEval-3, data-driven systems are better for timex detection, while rule-based ones get the best scores in normalisation.

### 2.2.2 TimeNorm

After the participation in TempEval-3 with ClearTK, in which the system got some of the best detection results, Bethard (2013b) proposed a new normalisation method by using synchronous context-free grammars[1]. This new method was designed to translate timexes into semantically compositional representations, and, as we have seen, these procedures were gathered later in the SCATE schema (Bethard and Parker, 2016). However, this system could not be evaluated on TempEval-3 because it did not perform the detection part.

Based on this annotation language, Laparra et al. (2018) developed a complete detection and normalisation system that could beat rule-based ones by means of a recurrent neural network approach. At the same time, it provided a deeper and more fine-grained interpretation of timexes than state-of-the-art systems. Xu et al. (2019) improved this approach using pre-trained contextualised character embeddings[2] (Akbik et al., 2018).

Despite the good results of the TimeNorm systems, the only adaptation to a language other than English is the Italian version of the SCFG approach (Mirza and Minard, 2015). We have seen that HeidelTime has been translated to more languages (including Spanish) and allows the adaptation to any language, and TIPSem has also been designed for Spanish tasks. In this context, our project aims at a comparable detection and normalisation system for English and Spanish that can evaluate the TimeNorm SCFG on TempEval-3.

[1] A detailed explanation of this method will be explained in Section 4.2.

[2] We will explain more on these embeddings in Section 3.4, as they are also used in our system.

# 3 Methodology

So far, we have commented on the main contributions done in temporal annotation and temporal information processing. In this section, we will explain how our system for timex detection and normalisation works and we will describe the resources that we used: namely, the TimeBank corpus, the TimeNorm system, the Flair framework and the evaluation methods.

## 3.1 Overview of our System

Our system is divided in two main parts, according to the different resources that were used and their separate evaluations:

- **TimeNorm timex normalisation:** This is the core of our project, in which we have adapted the original English system to normalise Spanish time expressions. It corresponds to the task performed in Bethard (2013b) and part of the task A in UzZaman et al. (2012).
- **Timex detection with Flair:** As a complement to the previous one, it extracts Spanish time expressions from raw text to be later normalised and identifies the type they belong to. It corresponds to part of the task A in UzZaman et al. (2012), and the whole task if combined with normalisation.

Let's describe first the TimeNorm normalisation. In this phase, we feed the TimeNorm system (Bethard, 2013b)[3] with isolated time expressions plus another timex that will work as a temporal anchor, that is, a fixed reference that will guide the normalisation of the others. In this project, we have used the document creation time (DCT) of the corpus text files as anchors. The system parses one of the time expressions according to the synchronous context-free grammar[4] and the anchor, and provides a TimeNorm representation. From this complex representation, the system can return the appropriate TimeML TIMEX3 value, which can be evaluated by comparing it to the gold standard one. The process is repeated for the rest of the provided time expressions and text files. Figure 1 shows how the TimeNorm normalisation works.

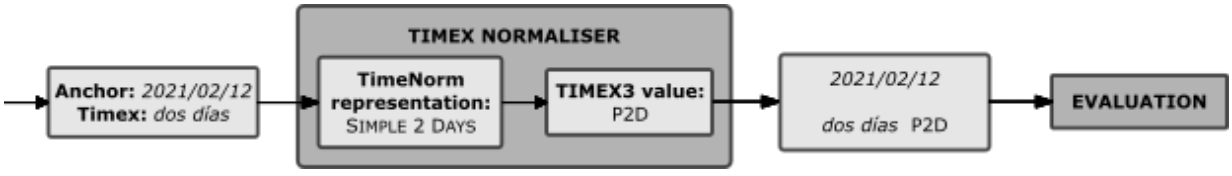


Figure 1: TimeNorm normalisation system.

[3] `https://github.com/clulab/timenorm`.

[4] See Section 4.2 to know how the TimeNorm SCFG works.

On the other hand, our timex detection system performs a sequence labelling task according to the IOB2 schema (Tjong Kim Sang, 2002; Tjong Kim Sang and De Meulder, 2003) by using Flair (Akbik et al., 2019a)[5]. During this phase, we apply the already trained system to a text file in order to find the possible timexes and identify their type, which can be TIME, DATE, DURATION or SET. The resulting time expressions are evaluated according to their span in the text and their type, and are then prepared to feed the TimeNorm normaliser, including the DCT of the file. The process can be repeated for the rest of the text files.

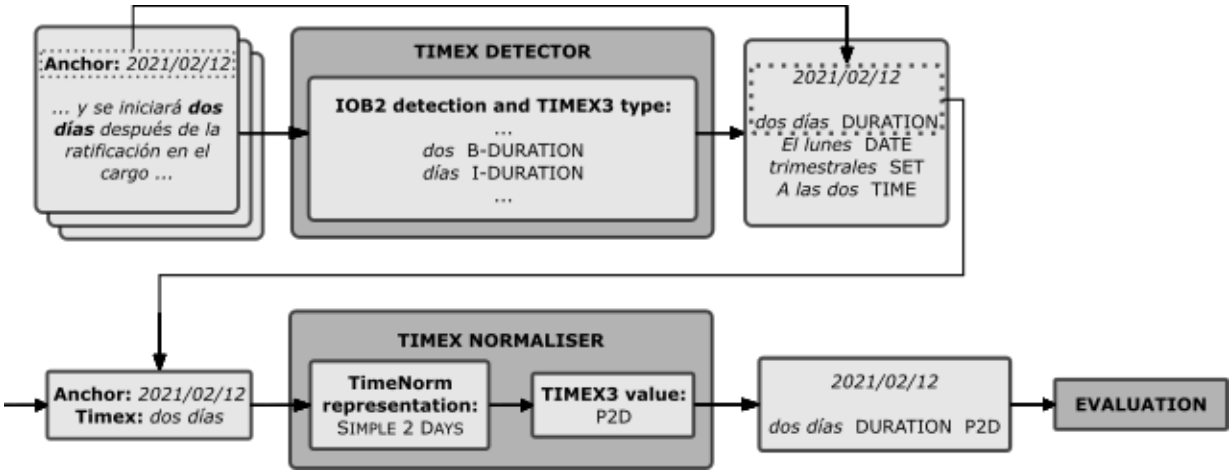


Figure 2: Overview of the system for detection and normalisation.

Figure 2 shows all the process if we do both timex detection and normalisation. For each text file, our detector takes the DCT as anchor, looks for possible time expressions regarding to the IOB2 schema and assigns them a timex type. For example, if we find the *dos días* timex (*two days*), the sequence is tagged word by word as belonging to one of the possible timex types, in this case, DURATION. The time expressions are sent one by one to the normaliser with the corresponding anchor (e.g. *2021/02/12*), where the system parses each timex by means of our Spanish grammar and generates its TimeNorm representation. In the case of *dos días*, it would be "SIMPLE 2 DAYS"[6]. From this representation, the normaliser provides the appropriate TIMEX3 value, for example, P2D.

## 3.2 The TimeBank Corpus

To train our system, we made use of the TimeBank corpus regarding the requirements of the TempEval-3 challenge. However, UzZaman et al. (2012) modified the corpora to make the formatting clearer, solve some incompatibilities and add missing expressions, events and relations, and that is the version of TimeBank we used both for English and Spanish.

[5] `https://github.com/flairNLP/flair`.

[6] See Section 4.4 to know how these representations are built.

The English timex detection and normalisation task in TempEval-3 required all TimeBank to train and a new "platinum" evaluation corpus was built. The Spanish task, on the contrary, had to be performed on the Spanish TimeBank both for training and evaluation. Table 1 shows the distribution of files, words and timexes in the corpora. The number of files and words come from TempEval-3, whereas the number of timexes, which include DCTs, are those that we found during normalisation alone. English timexes coincide with the numbers appearing in Bethard (2013b), but the Spanish ones do not match the amount of timexes according to Saurí and Badia (2012), neither with DCTs nor without them. This could be due to the revision and corrections done for TempEval-3.

| Corpus | # of files | # of words | # of timexes |
|---|---|---|---|
| TimeBank | 183 | 61,418 | 1,426 |
| TempEval-3 Eval | 20 | 6,375 | 158 |
| TimeBank-ES Train | 175 | 57,977 | 1,269 |
| TimeBank-ES Eval | 35 | 9,833 | 234 |

Table 1: Corpora for the TempEval-3 task A in English and Spanish.

The files are built on the XML format and they all present the document creation time, the text with the inserted tags and the links between the items. Time expressions, which are marked by the TIMEX3 tag, always have a unique identifier called "tid", a type and a value, among other optional features. The next example shows how timexes are defined in text with the three attributes:

1. ... y se iniciará <TIMEX3 tid="t5" type="DURATION" value="P2D">dos días </TIMEX3> después de la ratificación en el cargo...
   (... and it will begin *two days* after the ratification...)

We found some issues in the Spanish and English corpora that should be mentioned. Although the two of them work in this way, the file format is not the same, and the English one is much more heterogeneous across files. Typographic mistakes are more frequent in the English corpus, but we can also see, for example, that in the Spanish one whitespaces are systematically added between an opening parenthesis and a TIMEX3 tag. Moreover, we could think that some timexes can be set differently in the corpora, like in the example before where *después* (*after*) can form a TimeNorm timex with *dos días* but TimeML considers the adverb as a different kind of temporal element (Pustejovsky et al., 2005). Some of these cases will be discussed also in Section 5.

## 3.3 Timex Normalisation with TimeNorm

In the TimeNorm normalisation system we must first specify the timex language we want to use and the anchor, which is treated as a timex but must refer to a complete date (e.g. *2021/02/12*) or a complete date and time (e.g. *2021/02/12 12:45:13*). Once we have those

two things, we can provide a time expression. This timex passes through a tokenisation process to separate letters from non-letter characters (e.g. *one-night* as adjective) and consider, for example, a sequence of eight consecutive digits as a date (e.g. *20210212*). The Italian tokeniser also separates words referring to numbers (e.g. *quarantotto*, *forty-eight*), determiners (e.g. *queste*, *these*) and adjectives (e.g. *precedente*, *previous*) not to define rules for all the possible cases of numbers and inflections in the grammar. In Spanish, however, numbers are generally written compositionally and with whitespaces, so we decided to use the default tokeniser and represent all the inflection cases in the grammar[7].

After tokenisation, the timex is parsed on the TimeNorm synchronous context-free grammar. Here lies our main contribution, which is extensively described in Section 4. We must note that the TimeNorm grammar is designed to analyse time expressions but not to generate them, what allowed us to take concrete decisions during the construction phase. To parse with this grammar, TimeNorm uses an extended version of the CYK+ algorithm (Chappelier and Rajman, 1998) to address some of the cases that we need to use here. Finally, TimeNorm transforms the resulting representation in a TimeML normalisation regarding very simple conversion rules. In fact, many different TimeNorm representations can lead to the same TimeML annotation.

## 3.4 Timex Detection with Flair

Flair (Akbik et al., 2019a) is an easy-to-use NLP framework for sequence labelling, text classification and language modelling that has shown to beat analogous systems in NER, chunking and PoS tagging in English (Akbik et al., 2018), NER in German (Akbik et al., 2018) and NER in Spanish (Agerri and Rigau, 2020). It is designed to mix and match different embeddings (including pre-trained ones) through a unified interface that avoids the common engineering problems of doing this. Among the many kinds of embeddings that it supports, we can find classic word embeddings, such as GloVe (Pennington et al., 2014) or FastText (Bojanowski et al., 2017); Flair embeddings (Akbik et al., 2018) and pooled Flair embeddings (Akbik et al., 2019b), which will be briefly explained below; other word embeddings, like hierarchical character features (Lample et al., 2016), byte-pair embeddings (Heinzerling and Strube, 2018), ELMo (Peters et al., 2018) or BERT (Devlin et al., 2019); and stacked embeddings (Akbik et al., 2018), which are combinations of the previous ones.

We used Flair embeddings, also called contextual string embeddings, for our timex detection task. They are designed to gather the best attributes of the next types of embeddings:

- from **classical word embeddings**, the possibility to be pre-trained on very large corpora;

[7]See Section 4.3 to know more on this decision.

- from **character-level features**, that they are trained on task data, model words as sequences of characters and can better address rare and misspelled words and subword structures like inflection features; and
- from **contextualised word embeddings**, the capturing of word meaning in context and the generation of different embeddings for each sense.

Figure 3 shows how these Flair embeddings are generated and used to annotate a sentence like *George Washington was born*[8]. First, a pre-trained bidirectional character language model (in yellow) is fed with a sentence as a sequence of characters. For each of the words in the sentence, we retrieve a contextual embedding by taking the first and the last character cell states. Finally, these word embeddings are sent to a vanilla BiLSTM sequence labeller (in blue) to obtain, for example, that *George* and *Washington* are the beginning and the end respectively of the only named entity in the sentence.

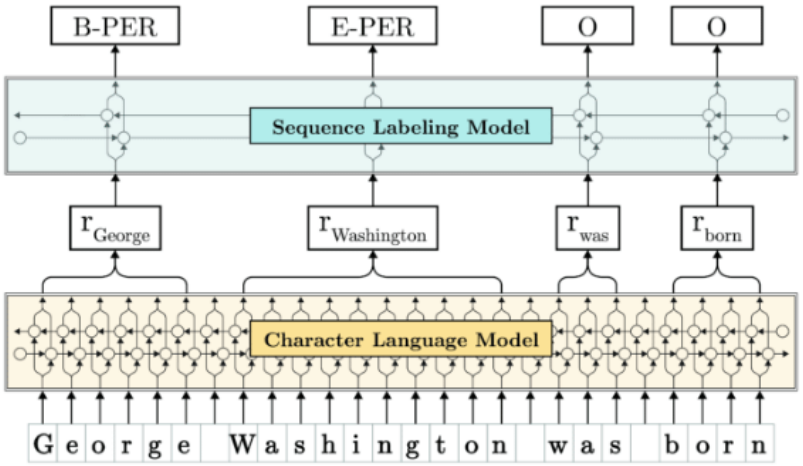


Figure 3: Flair embeddings.

Flair embeddings usually achieve a better performance if they are stacked with classic embeddings, so we tested them in combination with Spanish FastText embeddings, that are already available on the framework. We also tested the pooled version of these Flair embeddings, which are used to counteract their weakness to deal with rare words in underspecified contexts. To that end, the contextualised embeddings of each string are dynamically aggregated. With any of these embedding types, the pre-trained embeddings are trained on the corresponding corpus, which has been previously segmented, and a development set extracted from the train set is used during training to improve the model.

Our detection system takes sentences from a text and, by means of the Flair embeddings, assigns each word an IOB2 tag (Tjong Kim Sang, 2002; Tjong Kim Sang and

[8]This figure has been extracted from Akbik et al. (2018).

De Meulder, 2003). This sequence labelling schema indicates if a word matches the objective of the task or not. If it does, the first word is set to "B" (also if it is the only one) and the rest to "I", both followed by the timex type. If not, the word receives the "O" tag. This is why, in Figure 2, the example *dos días* is tagged as "*dos* B-DURATION", "*días* I-DURATION".

## 3.5 Evaluation Methods

As it can be seen from the overview of the system in Section 3.1, we did two different evaluations: one to evaluate normalisation, like the one performed by Bethard (2013b); and another one to evaluate the process from detection to normalisation, according to the TempEval-3 Task A in UzZaman et al. (2012). This last one, however, also includes the evaluation of the detection system alone.

### Evaluating timex normalisation

To evaluate the TimeNorm normalisation using our Spanish grammar, we had to write a new script that could normalise and test all the timexes from several files directly, since the system only provided the representations and the TIMEX3 values step by step. Previously, we had to parse the XML files from the corpus and get the timexes with their values in tabulated format to pass them to our new script. This method was used also during the normalisation development process to know how to improve our grammar before testing.

The evaluation in Bethard (2013b) corresponds simply to the ratio of correct normalisations provided by our system ($Sys$) over the total of the reference timexes ($Ref$), regardless of whether a normalisation is given or not, so we did the same. We must note that this evaluation also considered the normalisation of the DCTs. To compare our results with the original ones, we repeated the test also for English.

$$Accuracy = \frac{|Sys_{timex} \cap Ref_{timex}|}{|Ref_{timex}|} \quad (1)$$

### Evaluating timex detection and normalisation

Regarding the detection and normalisation process, TempEval-3 did provide an evaluation script, but we could not achieve reliable results with it[9]. For this reason, we tested our system following the TempEval-3 metrics, which should provide comparable results.

The metrics used in TempEval-3 for timex detection and normalisation to compare reference and system timexes are precision, recall and F1-score for strict match (e.g. *sunday morning* vs. *sunday morning*), relaxed match (e.g. *sunday morning* vs. *sunday*), type (e.g. DURATION vs. DURATION) and value (e.g. P2D vs. P2D). They are based on the metrics from Tjong Kim Sang (2002) and Tjong Kim Sang and De Meulder (2003), and can be calculated as specified below.

[9]We tried to contact the authors but did not get any answer.

**How many timexes are correctly identified:** Strict and relaxed matches ($M$) indicate if a timex has been correctly identified totally or in part (at least by one word), respectively. Their precision, recall and F1-score are computed in this way, where $Sys_{timex}$ contains the timexes detected by our system and $Ref_{timex}$ the reference timexes:

$$Precision_M = \frac{|Sys_{timex} \cap Ref_{timex}|}{|Sys_{timex}|} \tag{2}$$

$$Recall_M = \frac{|Sys_{timex} \cap Ref_{timex}|}{|Ref_{timex}|} \tag{3}$$

$$F1_M = \frac{2 * Precision_M * Recall_M}{Precision_M + Recall_M} \tag{4}$$

**How many timex attributes are correctly identified:** The possible attributes ($A$) for timexes are type and value, which are considered only if there is at least a relaxed match. The equations to calculate their precision, recall and F1-score are the next ones:

$$Precision_A = \frac{|\{\forall x | x \in (Sys_{timex} \cap Ref_{timex}) \wedge Sys_A(x) == Ref_A(x)\}|}{|Sys_{timex}|} \tag{5}$$

$$Recall_A = \frac{|\{\forall x | x \in (Sys_{timex} \cap Ref_{timex}) \wedge Sys_A(x) == Ref_A(x)\}|}{|Ref_{timex}|} \tag{6}$$

$$F1_A = \frac{2 * Precision_A * Recall_A}{Precision_A + Recall_A} \tag{7}$$

The final metrics that are used in TempEval-3 to evaluate timex detection and normalisation are strict F1, relaxed F1, relaxed precision, relaxed recall and value F1, as shown in Tables 12 and 13.

# 4 Towards Spanish TimeNorm

The proposal in Bethard (2013b) aims at a simpler and more powerful way to organize the rules for time normalisation. To this end, he defines the formal operations that deal with temporal elements and the synchronous rules to map time expressions to those operations, and then normalises the expressions via an extended CYK+ parsing algorithm. In this section we will first explain this synchronous context-free grammar formalism that has been the basis for our work, starting from a recapitulation of context-free grammars and explaining later the process of the TimeNorm system. Secondly, we will comment on the procedure to build our Spanish grammar. Lastly, we will describe the grammar itself, as well as the problems that arose and the decisions that were taken during the building process.

## 4.1 Recalling CFGs

A context-free grammar (CFG) is composed by a lexicon and a set of rules, which indicate how the words or symbols in the lexicon can be combined, or even how to embed some rules in others. In a formal way (Jurafsky and Martin, 2019), a context-free grammar $G$ is defined by the next four parameters:

- a set of non-terminal symbols $N$,
- a set of terminal symbols $\Sigma$,
- a set of rules of the form $A \rightarrow \beta$, where $A$ is a non-terminal and $\beta$ a string from the infinite set of strings $(\Sigma \cup N)*$, and
- a non-terminal start symbol $S$.

The next adapted extract of our grammar shows how a simple CFG works, from parent to children, where sequences in italics are terminals and those in brackets are non-terminals:

| PARENT | → | CHILDREN |
|---|---|---|
| [INTEGER] | → | [UNITS] \| [TENS] \| [TENS] *y* [UNITS] |
| [UNITS] | → | *dos* |
| [TENS] | → | *treinta* |

Table 2: A simple CFG.

Consider that an integer can be formed by the linguistic transcription of one or two digit number (for example, *thirty-two* instead of *32*), the only words (terminals) are *dos* (*two*), *treinta* (*thirty*) and *y* (*and*) and [INTEGER] is the start symbol. The first rule in the grammar indicates that we can create an [INTEGER] element with a) a [UNITS][10] element,

[10]Observe that this [UNITS] tag is not the same as the [UNIT] tag further below. Actually, [UNITS] is only used in this toy example and does not appear in our grammar.

b) a [TENS] element or c) a [TENS] element and a [UNITS] element joint by *y* terminal. The second rule states that there exists only one [UNITS] element, which is *dos*, whereas the last rule defines *treinta* as the only [TENS] element. Thus, if we follow the first rule through the other two, we can create three different integers: *dos*, *treinta* and *treinta y dos*, which, of course, are valid integer transcriptions in Spanish. In particular, to parse an expression like *treinta y dos* a tree like the one in Table 3 would be needed.

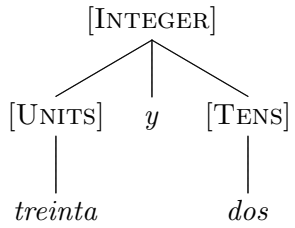


Table 3: Tree for *treinta y dos*.

## 4.2 The TimeNorm SCFG

However, this TimeNorm system requires a more complex structure called synchronous context-free grammar (SCFG). This kind of grammar builds two different trees simultaneously to parse an expression: the source tree for the expression itself (as in the toy example before), and the target tree for the temporal operators. Formally (Bethard, 2013b), the rules of the SCFG follow the structure $X \rightarrow (\mathbf{S}, \mathbf{T}, A)$, where

- $X$ corresponds to a non-terminal,
- $\mathbf{S}$ is the sequence of terminals and non-terminals that $X$ expands to in the source language,
- $\mathbf{T}$ is the sequence of terminals and non-terminals that $X$ expands to in the target language,
- and $A$ defines the alignment between the non-terminals in $\mathbf{S}$ and $\mathbf{T}$.

The example tree from Bethard (2013b) has been adapted to Spanish in Tables 4 and 5 to show the parsing of our SCFG for the expression *la semana del 6 de marzo* (*the week of March 6*).

As we can see, the source tree in Table 4 parses the natural language expression in a similar way to Table 3, while the target tree in Table 5 simultaneously parses the temporal operators according to the alignments marked in subscripts. The terminals in the source tree are still those expressions in italics, but in the target tree are now numeric values or temporal operators like FINDENCLOSING, PRESENT or DAYOFMONTH.

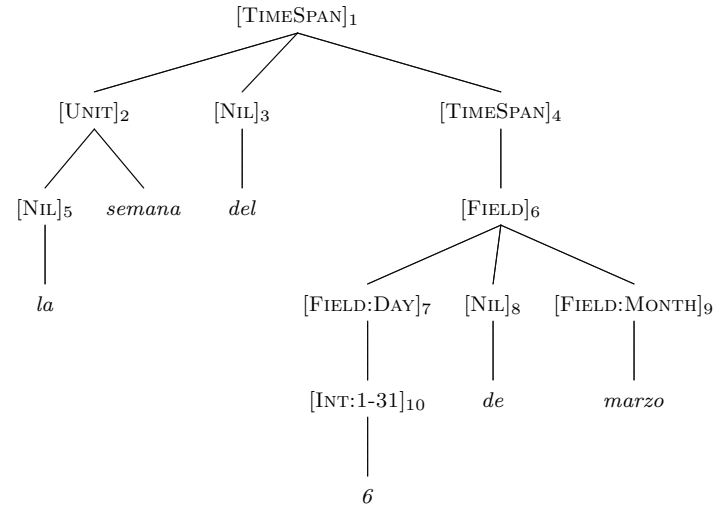


Table 4: Source tree for *La semana del 6 de marzo*.

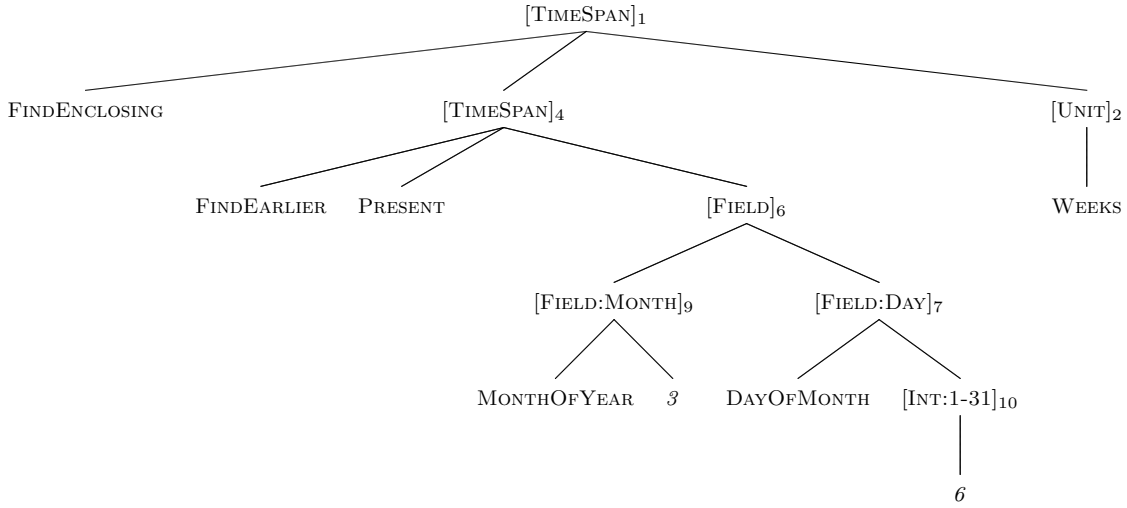


Table 5: Target tree for *La semana del 6 de marzo*.

On the other hand, both trees have the same non-terminals in order to indicate the alignments. However, there are two exceptions: a [Nil] non-terminal can appear only in the source tree to allow the terminal symbol below to be ignored, and a non-terminal may be used more times in the target tree than in the source tree if the target grammar requires to do so. If we follow the terminals of the target tree, we can find the target formal representation for the source expression *la semana del 6 de marzo*, which is "FindEnclosing(FindEarlier(Present, MonthOfYear 3, DayOfMonth 6), Weeks)". This interpretation corresponds to *the last week of March 6*, but there are other possible parses, like *the next week of March 6*. We will see later how disambiguation works.

Let's have a closer look to the target tree. The target non-terminals, which can be of type [TimeSpan], [Period], [Field], [Unit] or [Int], are created by combining terminals (integers or temporal operators) and other non-terminals. These combinations are intended to generate temporal objects according to the procedures defined by the temporal operators. For example, we can define *hoy* (*today*) as the temporal object FindEnclosing(Present, Days), where the instantaneous operator Present is expanded to the enclosing day, in our case, the document creation time. The basic rules that allow the possible combinations in the target grammar are shown in Table 6.

| PARENT | → | CHILD |
|---|---|---|
| [Int] | → | integer |
| [Unit] | → | unit |
| [Field] | → | field [Int] |
| [Field] | → | [Field]* |
| [Period] | → | Simple [Int] [Unit] |
| [Period] | → | Fractional [Int] [Int] [Unit] |
| [Period] | → | Unspecified [Unit] |
| [Period] | → | WithModifier [Period] modifier |
| [TimeSpan] | → | Past |
| [TimeSpan] | → | Present |
| [TimeSpan] | → | Future |
| [TimeSpan] | → | FindEarlier [TimeSpan] [Field] |
| [TimeSpan] | → | FindLater [TimeSpan] [Field] |
| [TimeSpan] | → | FindEnclosing [TimeSpan] [Unit] |
| [TimeSpan] | → | FindEnclosed [TimeSpan] [Field] |
| [TimeSpan] | → | StartAtEndOf [TimeSpan] [Period] |
| [TimeSpan] | → | EndAtStartOf [TimeSpan] [Period] |
| [TimeSpan] | → | MoveEarlier [TimeSpan] [Period] |
| [TimeSpan] | → | MoveLater [TimeSpan] [Period] |
| [TimeSpan] | → | WithModifier [TimeSpan] modifier |

Table 6: Basic rules allowed by the target grammar.

Among the rules in Table 6 there are four terminals that are not in small caps: "integer" is any integer number (in numeric form), "unit" is a `java.time.temporal.TemporalUnit` (for example, Seconds or Weeks), "field" is a `java.time.temporal.TemporalField` (like DayOfMonth or Century), and "modifier" is a TIMEX3 "mod" value in TimeML. The other terminals work as temporal operators that can act by themselves (like Present above, Past and Future) or alongside other terminals (for example, Simple, FindEarlier or StartAtEndOf) to create temporal objects.

So far we have described the TimeNorm SCFG in an abstract and simplified way. To know how it truly works, Table 7 shows an example of the real rules that parse the time

expression *la semana del 6 de marzo*[11].

| PARENT | → | CHILDREN |
|---|---|---|
| | | *la semana* |
| [UNIT:SINGULAR] | → | *semana* |
| | → | WEEKS |
| | | *6 de marzo* |
| [FV:MONTHOFYEAR] | → | *marzo* |
| | → | MONTHOFYEAR 3 |
| [FV:NUMERICDAYOFMONTH] | → | [INT:1-31] |
| | → | DAYOFMONTH [INT:1-31] |
| [FV:MONTHDAY] | → | [FV:NUMERICDAYOFMONTH] [FV:MONTHOFYEAR] |
| | → | [FV:MONTHOFYEAR] [FV:NUMERICDAYOFMONTH] |
| [FV:PARTIALYEARMONTHDAY] | → | [FV:MONTHDAY] |
| | → | [FV:MONTHDAY] |
| [FV:PARTIAL] | → | [FV:PARTIALYEARMONTHDAY] |
| | → | [FV:PARTIALYEARMONTHDAY] |
| [TS:REGULAR] | → | [FV:PARTIAL] |
| | → | FINDATORBEARLIER PRESENT [FV:PARTIAL] |
| | | Link the previous rules |
| [TS:REGULAR] | → | [UNIT:SINGULAR] [TS:REGULAR] |
| | → | FINDENCLOSING [TS:REGULAR] [UNIT:SINGULAR] |
| [TIMESPAN] | → | [TS:REGULAR] |
| | → | [TS:REGULAR] |

Table 7: Real rules to parse *la semana del 6 de marzo*.

Each rule indicates the transition from a parent node to its children, which are divided in two rows: the upper row corresponds to the source representation (e.g. *marzo*) and the lower row corresponds to the target representation (e. g. MONTHOFYEAR 3). If we follow the rules, we can see how to get from the terminals *semana*, *6* and *marzo* to the [TIMESPAN] root in the last row. We should note that here there are more steps than in the trees in Tables 4 and 5, since many tags (e.g. FIELDVALUE) are specified by using subtags (e.g. MONTHDAY), which were not referred in the basic target grammar of Table 6. Moreover, the rules show how words like *la*, *del* and *de* are not considered[12].

The TimeNorm system uses an extended version of the CYK+ algorithm (Chappelier and Rajman, 1998). This extension solves some of the problems of the original one, which does not address the mixing of terminals and non-terminals, the ignoring of words by means of the [NIL] non-terminal and the use of the [INT] non-terminal to match integers without having to manually list all the possibilities.

Furthermore, very often a timex can be parsed to various TimeNorm representations. In these cases, the system performs all the possible parses and selects one according to the following heuristic rules:

[11] We will sometimes use FV and TS instead of FIELDVALUE and TIMESPAN respectively to save space.
[12] See Section 4.4.1 to know more about expressions that can be ignored.

- prefer a [TIMESPAN] to a [PERIOD],
- prefer an earlier [TIMESPAN] to a later one, and
- prefer a [TIMESPAN] with QUARTERS granularity if the anchor time is also in QUARTERS (a common rule in TimeBank annotations).

Lastly, we did not modify the target grammar in order to maintain the integrity of the system. On the contrary, any adaptation of the original TimeNorm SCFG should be made on the source grammar instead (that is, the first row of each rule in Table 7) or by creating new rules that respect the target grammar symbols and syntax. This allows us to address forms in languages other than English (or to improve the original English version) by changing the grammaticality that is required to the input timexes. However, it should be noted that the target grammar imposes some restrictions that might not fit linguistic structures that are absent in English. Although this system was designed to work only with English expressions, if we aimed at a better and more cross-lingual performance, we would also need to adapt and extend the target grammar or directly the code of the TimeNorm system, as the authors of the Italian version did (Mirza and Minard, 2015).

## 4.3 Grammar Construction Approach

As it was already mentioned, our Spanish grammar has two precedents: the original English TimeNorm SCFG (Bethard, 2013b) and the Italian version (Mirza and Minard, 2015). Based on these two grammars, we have developed our own by adapting the English one to Spanish in a first place, and then comparing it to the Italian grammar in case it provided any improvement of the rules. The reason for doing this is simple: while the original one is the model that should be adapted because of it being the first and the best in terms of performance and efficiency, Italian linguistic structures are more similar to those in Spanish. Thus, if we needed to come up with a new way to write a rule for Spanish, we would make that first and then check whether the Italian solution was better.

At a smaller scale, we have tried to respect the original rules and symbols as much as possible. For example, we found out quite early that we could not directly translate the rules for number transcriptions, as we needed a new subtag to define words from *veinte* (*twenty*) to *veintinueve* (*twenty-nine*): in English there is no problem to build these expressions compositionally in the same way that we would create in Spanish the rest of the next two digit number transcriptions, like *treinta y nueve* (*thirty-nine*), but it is obvious that *veintinueve* and *treinta y nueve* do not share a common structure. To solve this problem, we had to create a new subtag called [INT:TWENTIES] inspired in the already existing [INT:TEENS] subtag, which accepted number transcriptions from *diez* (*ten*) to *diecinueve* (*nineteen*). This solution follows some guidelines that were adopted at the beginning of our grammar development for Spanish, and which consists of three main principles: a) Make the necessary changes in the source grammar whenever possible, b) Prefer a normative approach rather than a descriptive one and, c), build a linguistically coherent grammar rather than a task-oriented one.

**Make the necessary changes in the source grammar whenever possible**

This principle allows us not to preprocess the time expressions. The authors of the Italian system decided to do preprocessing on the input timexes, since they had to deal with accented letters, unified articles (and other determiners), articulated prepositions and concatenated numbers (Mirza and Minard, 2015) much more than we do in Spanish. That is the case in, for example, *lunedì* (*Monday*, word with an accented letter), *quest'estate* (*this summer*, word with a unified determiner), *dell'anno* (*of the year*, word with an articulated preposition) or *duemilaquattordici* (*two thousand fourteen*, a concatenated number), which in Spanish have a much simpler spelling: *lunes*, *este verano*, *del año* and *dos mil catorce*, respectively. In fact, we have observed that there is no need to adapt the TimeNorm tokenization system in order to work with accented letters or characters like "ñ".

Moreover, we decided to double the rules for expressions with two possible genders (whereas the Italian system mixed preprocessing and rule doubling) and also when plural marks were needed. Of course, because of the morphosyntactic features of Spanish such as gender marking, it was inevitable to build a larger and more complex grammar than the English one, although we managed to make it much simpler than the Italian proposal. The TimeNorm grammar structure, whose objective is to analyse time expressions (not to generate them), also allowed us to simplify some concrete rules by overwriting them, as well as the English and Italian versions did.

**Prefer a normative approach rather than a descriptive one**

By following this principle, our grammar can be simpler and more compact. This means that we have not tried to represent all the possibilities (what would be impossible and undesirable if we wanted to avoid unnecessary ambiguities and parsing complexity), but only those that we thought to be sufficiently representative of European Spanish. To that end, we consulted the official recommendations of the Ortografía de la lengua española (Real Academia Española, 2010) in case there was information about the matter. Nevertheless, we chose RAE just to have a coherent academic reference, since decisions were taken considering also the features of our grammar and the structure of the timexes, which are quite varied.

In particular, most of the doubts on this subject concerned the spelling of expressions. For example, we said that the transcriptions for numbers from thirty-one to one hundred (except for the tens) were written separated by *y* in Spanish (e.g. *treinta y nueve*), but we chose this spelling because it is the most common, even if writing them in a single word (*treintainueve*) is accepted (Real Academia Española, 2010).

**Build a linguistically coherent grammar rather than a task-oriented one**

We believe that this approach facilitates the adaptation of our grammar to different domains or applications. In order to have a first version of the grammar attending only to linguistic features, we built our grammar as we have described from beginning to end, before considering the corpus that was going to be used for training. However, we did have

by that time some training information from the English and Italian grammar, which were used as models.

Once the first version was finished, we started to evaluate the Spanish system on the training corpus, corrected or adapted the cases that could be fixed and added those that were not still considered, whenever the modifications were linguistically justified. Thus, if a possible change improved the training scores but had no linguistic sense, that change would not be made.

## 4.4 Building the Spanish TimeNorm Grammar

We have commented on some problems that arose during the building of the Spanish TimeNorm grammar to illustrate our procedure. We present hereafter a more detailed description of the cases considered in the grammar, the main differences between the three grammar versions (English, Italian and our own), the most relevant problems that we encountered and the solutions that we found to solve them.

### 4.4.1 Expressions that can be ignored

As we have already seen in Section 4.2, we can set some particles of time expressions to be ignored during the parsing process. That was the case in, for example, *la semana del 6 de marzo*, where *la*, *del* and *de* are not considered by the system. The decision to choose what particles would be in the list was not easy, since a small change would deeply modify the performance of the system, and having very few particles would force us to build a much larger grammar. Keeping these two things in mind, we decided to take advantage of the syntactic behaviour and low semantic content of the words that appeared in the English and Italian list, that is, determiners (e.g. *la*), prepositions (e.g. *de*) and combinations of both (e.g. *del*). Like in Italian, getting all of these particles in the list allows us not to double the rest of the rules more than necessary.

Even if we ignore words that are not ignored in the English grammar (for example, we do not consider *del* in the previous sentence, while the English grammar does), the syntactic features of the particles and the linguistic possibilities of the timexes are well represented in most of the rest of the grammar. Nevertheless, in those cases where there was some avoidable ambiguity between two or more rules, we clarified the disambiguation by including the corresponding particles in the rules. An exception is the determiner *una* (*a, an*), which was not included in the list because it coincides with the numeral adjective *una* (*one*) not only when it refers to anything (this is not a problem since the meaning is very similar, and occurs also with *un* and *uno*), but also when it refers to time, like in *la una en punto* (*one o'clock*).

During the building phase we found that the words *mismo* and *misma* (*same*, as in *the same day*) were frequent and temporally empty enough to be included in the list, although nothing similar was considered in the English nor Italian systems. On the contrary, the expressions *each of* and *more* from the English list were not adapted to Spanish, because they were better represented inside other rules. This happened also with the English *more*,

which in our grammar would have cancelled the rules that parse, for example, *más de un día* (*more than one day*) or *tres meses más* (*three months more*). To avoid this problem, we did not include *más* (*more*) in this group of expressions and wrote specific rules for the last cases.

### 4.4.2 Integers

We have to represent integer transcription to analyse time expressions like *las dos* (*two o'clock*), *doce meses* (*twelve months*) or *hace cien días* (*a hundred days ago*). Some comments on this matter have already been made: we need a rule for each of the numeral adjectives *un*, *uno* and *una* (*one*), and integer transcriptions are built compositionally[13] except for the transcriptions for numbers from 1 to 29. To these exceptions we add now the word *cien* (*a hundred*), which cannot be combined with one or two digit numbers and, as the others, is directly turned by the grammar into the integer it represents, 100.

The rest of the transcriptions work in this way: if we wanted to analyse the expression *dos mil veintiuno* (*two thousand and twenty-one*, *twenty twenty-one*) we would see that *dos* is parsed directly as 2 and *veintiuno* as 21. For the case of *mil*, the system would check if there are other consecutive number transcriptions and, in that case, of how many digits it is composed; as we do have one of two digits, *mil* would add a 0 between the previous and the next numbers, since it works compositionally. Thus, *dos mil veintiuno* is analysed as an [INT] element of value 2021. The rules that define this process are shown in Table 8.

| PARENT | → | CHILDREN |
|---:|---|---|
| [INT:1DIGIT] | → | *dos* |
| | → | 2 |
| [INT:TWENTIES] | → | *veintiuno* |
| | → | 2 1 |
| [INT:2DIGIT] | → | [INT:TWENTIES] |
| | → | [INT:TWENTIES] |
| [INT:THOUSAND3DIGIT] | → | *mil* [INT:2DIGIT] |
| | → | 0 [INT:2DIGIT] |
| [INT:4DIGIT] | → | [INT:1DIGIT] [INT:THOUSAND3DIGIT] |
| | → | [INT:1DIGIT] [INT:THOUSAND3DIGIT] |
| [INT] | → | [INT:4DIGIT] |
| | → | [INT:4DIGIT] |

Table 8: Rules that parse *dos mil veintiuno*.

Apart from *un*, *uno*, *una*, in Spanish there is another case of gender marking in cardinal number transcriptions, the hundreds: for example, *doscientos* and *doscientas* (*two hundred*). Moreover, we can see here that we lose some degree of compositionality, as in English we can say *dos cientos* (*two hundred*) and the Italian system accepts *due cento* after preprocessing *duecento*. As we mentioned before, Italian presents more spelling constraints than Spanish, what makes their system need both doubling of rules and preprocessing.

[13] This works similarly to Figure 2, but with target rules and in a more complex way.

On the other hand, the only ordinal number to represent a day of a month that we have included is *primero* (*first*), because in Spanish month days are always written in numeric form except for this infrequent option (Real Academia Española, 2010). In the Italian grammar there are no rules for ordinal numbers.

### 4.4.3 Temporal units

Temporal units are [UNIT] elements, which transform singular or plural time expressions like *segundo* (*second*), *semanas* (*weeks*) or *trimestre* (*quarter*) in operators like SECONDS, WEEKS or QUARTERYEARS respectively, and will later be used to create temporal objects. This is one of the cases where English structures constraint our definition possibilities. For example, in Spanish *cuatrimestre* (*four-month period*) and *semestre* (*semester*) are usually used, but there are no such `java.time.temporal.TemporalUnit` operators. In consequence, we cannot get a formal representation that leads to a correct normalisation according to their TimeML classification.

### 4.4.4 Periods

Periods are [PERIOD] elements, which are built by combining [INT] and [UNIT] elements (*tres años*, *three years*), creating unspecified [UNIT] elements (*un rato*, *a while*) or modifying previous [PERIOD] elements (*unas cinco horas*, *about five hours*), as shown in Table 6. The elements of this class are the only ones that can remain unanchored, that is, they are not assigned to any concrete day and time. For example, *tres años* constructs any three year period by itself (unanchored), but may also be used to build a more complex [TIMESPAN] element (anchored) like the one for *hace tres años* (*three years ago*), that refers to a day three years before a concrete day. TimeNorm considers different types of periods:

- **simple periods** (e.g. *tres años*, *three years*),
- **periods with operations** (e.g. *dos años y tres meses*, *two years and three months*),
- **unspecified periods** (e.g. *un tiempo*, *a while*), and
- **periods with modifiers** (e.g. *unas cinco horas*, *about five hours*).

In addition to those period types, we can also build **period sets**, like *diariamente* (*daily*).

#### Simple periods

Simple periods are the ones that are built by combining [INT] and [UNIT] elements, like *tres años* (*three years*). Following the Italian grammar, we added period representations for the multiple unit timexes that were not considered among the units from Section 4.4.3, like *semestre* (*semester*) as "SIMPLE 6 MONTHS", *trienios* (*triennia*, *three-year periods*)

as "Simple 3 Years" or *milenaria* (*millenary*, *thousand-year-old*) as "Simple 1000 Years". As it can be seen, those expressions are both nouns or adjectives and doubled in gender and number[14]. In this way, even if the representation is not as complete and specific as in *trimestre* (*quarter*, QuarterYears) and might not lead to the correct normalisation according to the corpus, at least it provides a minimally appropriate solution.

The English grammar accepts the possibility of adding *first* before "Simple [Int] [Unit]" forms, like in *first three years*, and the Italian one includes many other options. The idea is to ignore the additions and treat the expression as a "Simple [Int] [Unit]" period, as there is no specific normalisation tagging for this kind of sequences. Based on them, we created rules for expressions starting with, for example, *primer* (*first*), *segundo* (*second*) or *último* (*last*) in various combinations and gender and number forms. In this way, *los primeros nueve meses* (*the first nine months*), *estos nueve meses* (*these nine months*) or *los nueve últimos meses*[15] (*the last nine months*) would receive the same normalisation ("Simple 9 Months"), as well as *la primera semana* (*the first week*), *la tercera semana* (*the third week*) or *una semana más* (*one week more*) ("Simple Weeks").

### Periods with operations

We adapted two simple operations affecting units to generate a bit more complex periods. The first, appearing in the Italian grammar, is the sum of already created periods, which in Spanish expressions is constructed by inserting the *y* conjunction between them: for example, *dos años y tres meses* (*two years and three months*). On the other hand, we translated the rules from the English grammar to build fractional periods and corrected the Italian proposals, in order to transform, for example, *medio año* (*half a year*) in "Fractional (Int 1) (Int 2) Years" and normalise it as six months. The same can be applied to *dos medias horas* (*two half-hours*, to "Fractional 2 (Int 2) Hours", to one hour) or *tres años y medio* (*three years and a half*, to "Fractional 3 (Int 1) (Int 2) Years", to three years and six months).

### Unspecified periods

To parse time expressions like *un tiempo* (*a while*), TimeNorm provides the Unspecified operator. For each noun that we considered to be needed among unspecified periods, we doubled the rules to cover different senses. For example, with *tiempo* we can also build *largo tiempo* (*long time*), *poco tiempo* (*little time*) or *cierto tiempo* (*some time*). Based on the original grammar and the additions of the Italian one, we also included rules to transform unspecified units in periods, like *alguna semana* (*some week*), *varios meses* (*several months*) or just *años* (*years*) are all turned into "Unspecified Unspecified".

[14]Some of this kind of adjectives are not here but in "Period sets", since they show a frequency sense more than the others.

[15]Note that here the added word is between the integer and the unit, unlike the other cases.

#### Periods with modifiers

The last type of periods are those that modify other periods. For example, if we add *unas* (*about*) before *cinco horas* (which lead to a period of five hours), the result is "WithModifier (Simple 5 Hours) Approx". Apart from the Approx operator, we also use LessThan (e.g. *menos de un año*, *less than a year*), MoreThan (e.g. *superior a tres meses*, *of more than three months*), EqualOrMore (e.g. *año y medio como mínimo*, *a year and a half at least*) and EqualOrLess (e.g. *no más de treinta minutos*, *no more than thirty minutes*). Nevertheless, at the end, the normalisation of timexes like *unas cinco horas* is the same as the one for *cinco horas* alone, since TimeML does not reward this kind of precision and allows it only as an optional tag. In any case, TimeNorm includes this specificity in their aim of improving time representations (Bethard and Parker, 2016).

#### Period sets

Unlike the adjectives from "Simple periods", there are other adjectives and adverbs referred to units that show a frequency sense. This is the case for *diariamente* (*daily*) or *bimestrales* (*two-month*), which represent repeated times and are turned into the next period sets: "WithQuantifier (Period Simple 1 Days) Every" and "WithQuantifier (Period Simple 2 Months) Every", respectively. Apart from these one-word timexes, TimeNorm accepts expressions built with, for example, *cada* (*each*), *una vez por* (*once each*) or *todas* (*every*), and a time expression leading to a period, like *semana* (*week*), which results in *cada semana*, *una vez por semana* or *todas las semanas*. Again, all these forms are normalised in the same way as *una semana* (*one week*) at the end. Nevertheless, we can think of the possibility to accept expressions like *dos veces por semana* (*twice a week*), which has not been implemented yet.

### 4.4.5 Field values

Temporal fields are `java.time.temporal.TemporalField` basic operators (for example, AMPMOfDay, DayOfWeek or MonthOfYear) and can build field values. These field values are [FieldValue] elements, which transform timexes like *de la mañana* (*in the morning*), *jueves* (*Thursday*) or *Nochebuena* (*Christmas Eve*) in operators like "AMPMOfDay 0", "DayOfWeek 4" or "(FV MonthOfYear 12) (FV DayOfMonth 24)" respectively. As we can observe, field values combine basic field operators and integers that indicate which concrete field operator the time expression corresponds to, and can even combine different field values. But, unlike periods, in TimeNorm they are always anchored through a bigger [TimeSpan] element, which can be undesirable when the normalisation of a timex from the corpus remains unanchored. A TimeNorm grammar should include various types of field values:

- **parts of a day** (e.g. *noche*, *night*) and **time** (e.g. *16:32:27*),
- **days, months, quarters and seasons** (e.g. *lunes*, *Monday*),

- **dates** (e.g. *16/02/2021*), and
- **unspecified field values** (e.g. *otoños*, *autumns*).

### Parts of a day and time

These correspond to day sections described by AMPMOfDay, MorningOfDay or HourOfDay, for example. As these concepts are highly dependent on cultural features, it is not easy to match Spanish time expressions with the given English-based operators. This is the case for *mediodía* (*midday*), where we can argue whether it corresponds exactly to "twelve noon" (a time) or is rather "the period for lunch" (a part of a day). We finally opted for including both, the first as "(FV HourOfDay 12) (FV MinuteOfHour 0)" and the second as "AfternoonOfDay 1" (1 for positive). Although the Italian grammar includes the expression *ora di pranzo* (*time for lunch*) also as AfternoonOfDay 1, it has no other meaning. For both languages, we would probably need new adapted configurations of TimeNorm to achieve a better representation that can besides fit the TimeML normalisations, for example, to consider an explicit normalisation tag as the existing TimeML one for *mediodía* as part of a day.

Another tricky Spanish time expression is *mañana*, which can either refer to *tomorrow* and *morning*. To distinguish them, we took advantage of their different morphosyntactic relations, since *mañana* (*morning*) goes generally after a determiner, for example in *la mañana* (*the morning*) or *esta mañana* (*this morning*), while *mañana* (*tomorrow*) is usually alone as an adverb.

It must be noted that some of the timexes that lead to a part of a day operator are adjectives, like in the Italian grammar, and thus need to be doubled both in gender and number. For example, *nocturno*, *nocturna*, *nocturnos* and *nocturnas* (*night*, as an adjective) are linked to NightOfDay 1. In the English grammar, these adjectives are represented by adjectivised nouns.

Time operators are built in various ways: from single words (e.g. *mediodía*, in English *twelve noon*), from numbered hours (e.g. *16 h* or *4 p.m.*), hours and minutes (e.g. *16.32*), and hours, minutes and seconds (e.g. *16:32:27*), from times in letters (e.g. *las cuatro y media*, *half past four*), etc. To accept these Spanish constructions, we have had to modify and add some rules, but some time expressions present a too heterogeneous form to have their own rule.

During the training phase we observed some problems concerning time: for example, common Spanish expressions like *las 12 del mediodía* (literally *twelve in the noon*) or *las 12 de la noche* (literally *12 at midnight*) cannot be normalised as the operator HourOfAMPM does not accept 12 as integer. On the other hand, particles like *hora local* (*local time*) or *GMT* (meaning *Greenwich Mean Time*) often appeared in the corpus at the end of expressions referred to time, so we included these forms. Nonetheless, TimeNorm does not apply the TimeML normalisation tag for GMT times except for the anchors (they are defined in another way), which generates mistakes both in Spanish and English.

Moreover, we find here the problem of TimeNorm that makes field values be always anchored, since they belong to [TimeSpan] elements. The TimeNorm system directly assigns all the parts of a day and time expressions to a concrete day, while in some cases from the corpus they remain unanchored. For example, even if in a given context *tarde* (*evening*) is to be considered as an (abstract) period, in TimeNorm it is treated as a (concrete) time span.

### Days, months, quarters and seasons

Simple timexes referring to weekdays (e.g. *lunes*, *Monday*), months (e.g. *febrero*, *February*), quarters (e.g. *primer trimestre*, *first quarter*) and seasons (e.g. *primavera*, *spring*) also correspond to the form "DayOfWeek [Int]", "MonthOfYear [Int]", and so on. For some of these constructions we have included abbreviations, like *ju* for *jueves* (*Thursday*) or *ago* for *agosto* (August). Months can be written in numeric form to allow structures like *16/02/2021*, while month days are all written in numeric form except for the possibility of *primero de* [mes] ([month] *first*), as we already commented in Section 4.4.2. We have also mentioned the absence of a formal definition for month sets other than *trimestre* (*period*). For this reason, we have assigned the time expressions *primer trimestre* (*first quarter*) to *cuarto trimestre (fourth quarter)* to the form "QuarterOfYear [Int]" and added *último trimestre* (*last quarter*) as "QuarterOfYear 4", which generates some ambiguity with later constructions using *último* (*last*).

### Dates

Dates are built by combining days, months and years, although there is no need to indicate them all explicitly. For example, *16/02/2021*, *16 de febrero de 2021* and *martes, 16 de febrero del año 2021* are normalised to the same day, but also *16 de febrero* if the anchor day is after that date, *el 16* if the anchor is set in February or *el martes* if the anchor is set in the same week. In the case of *el 16*, this is an implementation that we did in the Spanish grammar, which was not in the English version and did not work well in the Italian one. However, these partial times are highly dependent on the heuristic rules for disambiguation and may sometimes fail, whereas absolute times are always exact.

Continuing with partial time combinations, the original English grammar accepts two digit years alongside a month and a day and assigns it to the last year with that end, like *February 16 21* is assigned to the 16th February 1921[16]. Nevertheless, it cannot parse a two digit year alone, so we implemented the rules to achieve it: in our Spanish grammar, *16 de febrero del 21* is correctly parsed to the 16th February 2021[17], but also *el año 21* to 2021 or *febrero del 21* to the whole month of February 2021. In this sense, we have also added a rule to parse *primavera de 2021* and *primavera del 21* (*spring of 2021*), which

[16]Other combinations result in other normalisations. For example, *16 February 21* is normalised as the 21st February 2016. It must be noted that this structure corresponds to the American English date form, whereas the British English form would normalise it in the same way the Spanish grammar does.

[17]This is in case the anchor is set on or after that date. If the anchor were set to the 2nd February 2021, the year would be 1921.

were impossible in the English grammar. The Italian grammar does not address these cases either.

In addition to the normal date structure, there are some specially named days that are directly normalised to a concrete day and month, like *Nochebuena* (*Christmas Eve*) or *Día de la Madre* (*Mother's Day*). For this last, for example, we have writen a rule that takes the first Sunday of May: "(FV DayOfWeek 7) (FV MonthOfYear 5) (FV AlignedWeekOfTheMonth 1)". As before, it would take the last date before the anchor unless a year is specified, like in *el Día de la Madre de 2021*. On the other hand, we have modified and added some of the most important specially named days in Spain, even if there are more that could be included.

#### Unspecified field values

Similarly to unspecified periods, we can consider some of the previous field values to work unspecifically inside time spans. This allows us to accept time expressions that refer to repeated times, although only for some plural nouns, since the rest are treated as partial. For example, we set *noches* (*nights*) or *otoños* (*autumns*) as unspecified, so we can transform expressions like *noches de jueves* (*Thursday nights*) and *los últimos tres otoños* (*the last three autumns*) in time span sets[18]. The expressions that lead to unspecified field values are parts of a day, year seasons, year quarters and the concrete expression *fines de semana* (*weekends*).

### 4.4.6 Time spans

Time spans are [TimeSpan] elements, which can be built by themselves (e.g. *antaño*, *in the past*) or based on integers (e.g. *los años 70*, *the 70's*), units (*esta semana*, *this year*), periods (e.g. *los últimos tres meses*, *the last three months*), field values (e.g. *esta tarde*, *this evening*), combinations of them (e.g. *verano de este año*, *summer this year*), or combinations with other time spans (e.g. *el día antes del pasado martes*, *the day before the last Tuesday*). Our previous example *la semana del 6 de marzo* (*the week of March 6*) was parsed from a time span root through another time span, several field values and a unit, as we can see in Table 7. Time spans can be divided in many different types:

- **solo time spans** (e.g. *en la actualidad*, *currently*),
- **date and time**, **decades** and **centuries** (e.g. *siglo XVII*, *17th century*),
- **time spans** for **past**, **present** and **future** (e.g. *el día después*, *the day after*),
- **time spans with demonstratives** (e.g. *esos años*, *those years*),
- **composed time spans** (e.g. *la noche de ayer*, *yesterday night*), and
- **modified time spans** (e.g. *a mediados de mes*, *at mid-month*).

[18] See the corresponding part in Section 4.4.6.

Similarly to periods, time spans can also build **time span sets**, like *los próximos inviernos* (*the next winters*).

### Solo time spans

Solo time spans are those that cannot be combined and are directly defined. For example, *en la actualidad* (*currently*) is turned into the [TimeSpan] element "Simple Present" without any compositional step. These time spans can only be "Simple Past", "Simple Present" or "Simple Future". Despite this simplicity, we found some problems throughout the building process.

Like in the Italian grammar, we had to accept as solo time spans expressions that also generate other kinds of time spans. For example, *Anterior* (*last*) is in our grammar but not in the original one, because this kind of solo adjectives are quite frequent in Spanish (*el anterior*, *the last one*). However, as these expressions are directly defined as [TimeSpan] elements, there is no possible ambiguity with the coincidences in other types.

We do find problems when we try to assign "Past", "Present" or "Future" to some timexes. Words like *entonces* (*then*), *aún* (*yet*) and *ya* (*already*, *now*) behave as referents of the verbal tense of their sentence, both in Spanish and English. For example, we can say *ya se hizo* (*it was already done*, *ya* is a past moment), *ya está hecho* (*it is already done*, *ya* is in the present) and *ya se hará* (*it will be done then*, *ya* is a future moment). Thus, depending on the context, they can refer to any of the three tenses[19]. In the English grammar, *yet* and *already* (*then* does not appear anywhere) are set to "Present". To make our decision, we checked during the training phase which was the most frequent tense for those expressions, and we found that *entonces* should turn into "Past". However, *aún* and *ya*, although quite frequent, were never tagged as TIMEX3 on their own. As a solo time span would affect the rest of the parse when combined with others, we did not include those two expressions, even if they show a clear temporal meaning.

For other similar expressions, we followed this same strategy. Concretely, those with *aquel*, *aquella*, *aquellos*, *aquellas* and *ese*, *esa*, *esos*, *esas* (*that*, *those*) are generally assigned to the past, while the expressions with *este*, *esta*, *estos*, *estas* (*this*, *these*) correspond more likely to the present.

### Date and time, decades and centuries

When we described how to construct dates in Section 4.4.5, we said that absolute dates (e.g. *16/02/2021*) were directly normalised to their correct value, while partial dates (e.g. *16 de febrero*, *6th February*) were set to the last possible value the expression could have. The reason for this is a rule that states that partial values generate [TimeSpan] elements of the form "FindAtOrEarlier Present [FV:Partial]" or "FindLater Present [FV:Partial]", which means that two possible values are given: the one from the present field value or earlier, or the next one. Note that here, "present" means the anchor time

[19] *Entonces* can refer to a past or a future moment but not to the present, since it imposes some sort of distance from the referred time to the speaker's time.

span. Before this ambiguity, the system uses the heuristic rule that commands that past time spans prevail. This happens not only with dates, but also with time or field values like *fin de semana* (*weekend*).

Decades and centuries are built in a similar way to partial field-value time spans. The general way to parse *los años treinta* (*the thirties*), *los años cuarenta* (*the forties*), and so on is to transform them in "FindAtOrEarlier Present (FV DecadeOfCentury [Int:Tens])", which takes the present or last decade of a century that matches the integer. As *veinte* had a different definition from the other tens, it has its own rule to parse *los años veinte* (*the twenties*), as well as the decades written with numbers (e.g. *los 20*, *the 20's*) and the special case of *los dos mil* (*the two thausands*) and *los 2000* (*the 2000's*).

On the other hand, centuries in Spanish are written in roman numbers, so we had to write all of them explicitly. For example, *siglo XXI* (*21st century*) is turned into "EndAtStartOf (TS FindAbsolute (FV Century 21)) (Period Simple 1 Centuries)", which matches the time span that ends when the absolute 21st century starts and which lasts the period of one century. Nonetheless, with the current TimeNorm system we can only work with centuries and years after Christ.

**Time spans for past, present and future**

The next time spans, like solo ones, are referred to the past, the present or the future, but, unlike those, are built compositionally. For example, *hoy* (*today*) is transformed in "FindEnclosing Present Days" (the time span of a day enclosed in the present, that is, the anchor time span), *ayer* (*yesterday*) in "EndAtStartOf (TS FindEnclosing Present Days) (Period Simple 1 Days)" (a one-day long time span that ends when "today" starts) and *anoche* (*yesterday night*) in "FindEarlier Present (FV NightOfDay 1)" (the time span of the night before the present). Apart from this kind of "basic" expressions, most of the timexes considered here are compositions of a unit, a partial field value, a period or a time span (or various of these) with an adjective of temporal meaning, like *anterior* (*previous*), *actual* (*current*) or *próximo* (*next*), or with a prepositional or adverbial construction, like *al cabo de* (*after*), or both together.

**Time spans for the past:** Among the past time expressions, we find combinations of adjectives with partial field values (e.g. *la pasada tarde*, *the last evening*) and periods (e.g. *aquellos tres años*, *those three years*), prepositions or adverbs with periods (e.g. *hace dos meses*, *two months ago*), and bigger constructions built on periods and time spans (e.g. *el día anterior al martes*, *the day before Tuesday*) or on partial field values and periods (e.g. *el otoño del año pasado*, *autumn last year*).

In the case of prepositions or adverbs with periods, which are formed as "*hace* [Period]", "[Period] *antes*" and "[Period] *atrás*" ("[Period] *ago*", "[Period] *before*"), there are two possible interpretations: for example, we can think that *hace dos meses* refers to the month two months ago or the exact day two months ago from the anchor. TimeNorm considers both options and selects the first one preferably.

On the other hand, we added the possibility to accept *ayer, 16 de enero* (*yesterday, 16th January*) or *ayer, miércoles* (*yesterday, Wednesday*), which was not possible in the

other grammars, but it normalises correctly only if the date is right. Moreover, remember that when we accept expressions with undetermined periods (e.g. *las últimas semanas*, *the last weeks*) the matched period is that one from the anchor back, until the last possible moment that the system can consider, which sometimes contradicts the meaning of the timex.

**Time spans for the present:** Present time expressions include also the form *hoy, 16 de enero* (*today, 16th January*) or *hoy, miércoles* (*today, Wednesday*), apart from *hoy* and other simple expressions. This time, we have combined units with determiners (e.g. *la semana*, *the week*) or adjectives (e.g. *la semana actual*, *the current week*), partial field values with determiners (e.g. *esta tarde*, *this evening*) or adjectives (e.g. *el presente trimestre*, *the present quarter*), and partial field values with units through determiners (e.g. *enero de este año*, *January this year*). The English and Italian grammar accepted also expressions with *that* as present, but at least in Spanish it does not make much sense since it represents a significant distance between the speaker's time and the referred time.

**Time spans for the future:** Finally, future time expressions cover simple forms like *mañana* (*tomorrow*) and its compounds (e.g. *mañana, 16 de enero*), and compositions of adjectives and others with partial field values (e.g. *el lunes que viene*, *next Monday*), periods (e.g. *las tres semanas siguientes*, *the next three weeks*) and both partial field values and periods (e.g. *lunes de la semana que viene*, *Monday next week*). Remember that we have no problem with *mañana* (*morning/tomorrow*), since the field value meaning takes the word only if it goes after a determiner.

Similarly to past timexes, TimeNorm considers two interpretations for expressions with periods: one referring to the corresponding period in the future and another referring to the exact day after the period. However, these constructions are easier than past ones to be distinguished according to the presence of determiners, so we made explicit these morphosyntactic differences to avoid any unnecessary ambiguity. Another coincidence is the fact that expressions with undetermined periods (e.g. *las próximas semanas*, *the next weeks*), are normalised as the period from the anchor to the last possible moment the system can take.

On the other hand, we found two problems on future timexes: first, units smaller than DAYS cannot be used inside periods to refer to the future (e.g. *la próxima hora*, *the next hour*); and second, parts-of-a-day field values (e.g. *la noche siguiente*, *the next night*) are normalised to that field value in the anchor day instead of the next day. The latter is correct according to the system, but does not match the human understanding of the time expression.

### Time spans with demonstratives

Some of the previous time expressions presented demonstratives in their structure. To generalise other cases, several rules have assigned the tag [TS:ALLOWSTHIS] instead of other kinds of time spans to indicate that those expressions can have a demonstrative before. The possible demonstratives are: *este*, *esta*, *estos*, *estas* (*this*, *these*); *ese*, *esa*, *esos*, *esas* (*that*, *those*, temporally close); and *aquel*, *aquella*, *aquellos*, *aquellas* (*that*, *those*,

temporally far).

Contrarily to the cases where we needed to indicate directly which demonstrative corresponds to what tense (e.g. *esta semana*, *this week*), these other time spans are normalised correctly without regard to the demonstrative, since the tense is already marked by other words. As the expressions that can have a demonstrative are not the same in English, Italian and Spanish, we needed to add or remove some [TS:AllowsThis] tags.

### Composed time spans

We can combine the previous time spans with units or partial field values to refer to more specific time spans. Some examples of composed time spans are *la semana del 6 de marzo* (*the week of March 6*), *la tarde de anteayer* (*the evening of the day before yesterday*) or *invierno del 94* (*winter of 1994*). To match any combination, the rules have been simplified to the extent possible.

### Modified time spans

Apart from the modifiers for periods in Section 4.4.4, there are others that can be used to create time spans: those applied to units, like Start (e.g. *a principios de mes*, *at the beginning of the month*), Mid (e.g. *a mediados de mes*, *at mid-month*) and End (e.g. *a últimos de mes*, *at the end of the month*); and those applied to time spans, like Before (e.g. *antes del lunes*, *before Monday*), After (e.g. *después del lunes*, *after Monday*), OnOrBefore (e.g. *hasta el lunes*, *until Monday*), OnOrAfter (e.g. *desde el lunes*, *from/since Monday*) and, again, Approx (e.g. *sobre el lunes*, *around Monday*). Like with periods, these modifiers do not change the normalisation of the timex, which is the same as for the modified unit or time span. They just provide a more specific TimeNorm representation that is not rewarded in the TimeML normalisation.

### Time span sets

Finally, we can create time span sets from partial and unspecified field values, as we did with period sets in Section 4.4.4. If we remember that unspecified field values are plural versions of some partial field values, we can see that *cada otoño* (*every autumn*) is built on a partial and *otoños* (*autumns*) or *esos otoños* (*those autumns*) on an unspecified one. The sets created on unspecified field values include demonstratives (like the last) and the words *últimos*, *últimas* (*last*) and *próximos*, *próximas* (*next*) in different combinations and with or without integers (e.g. *los últimos tres veranos*, *the last three summers*). However, instead of being normalised to any tense, they remain unspecified. There is also the possibility of combining an unspecified with a partial, like *noches de viernes* (*Friday nights*). Lastly, we have included the option to add *todos*, *todas* (*every, all*) to time span sets, so we can accept expressions like *todos los últimos inviernos* (*every last winter*) or *todas esas mañanas* (*all those mornings*).

In this last section, we have explained the different types of elements that conform TimeNorm grammars. To conclude, we summarise the main difficulties and problems that we found during this building phase:

- Language typology affects the way in which we construct a grammar. Spanish, as it is more inflected than English, needs more rules to adapt, for example, gender doubling and morphosyntactic flexibility.
- The cultural conceptions that are denoted by timexes are also difficult to represent in a logical way, and even more if we want to create only one rule to avoid ambiguities. Furthermore, some cultural conceptions related to time cannot be represented with the English-based TimeNorm schemes (e.g. *las doce de la noche*, *midnight*).
- The expressions that we have chosen to be ignored can interact with other rules, so they must be selected carefully.
- More common temporal units (e.g. *semestre*, *semester*) should be implemented to be able to provide them a correct normalisation.
- Field values are always assigned to a day in TimeNorm, instead of having the possibility to remain abstractly like periods do and TimeML allows.
- There are time spans whose reference depend on the context and cannot be disambiguated only with rules (e.g. *ya*, *already, now*).
- Disambiguation is always difficult to deal with for any grammar. In some cases we have solved the ambiguities by analysing the morphosyntactic features of a timex and building rules in accordance to them.
- Some of our date normalisations are incorrect because of the complete dependence on the DCT as anchor.
- A TimeNorm normalisation can be adequate and still not coincide with TimeBank (e.g. *media hora*, *half an hour*, to "PT30M" instead of "PT0.5H").

# 5 Experimental Results

In the last section, we described the TimeNorm SCFG and the main element types that constitute our grammar. We also indicated the biggest problems that we found during the building process, some of which even transcend it. After this building phase, we evaluated our newly created TimeNorm Spanish grammar on timex normalisation, according to Bethard (2013b), and our Flair-TimeNorm system on timex detection and normalisation, following UzZaman et al. (2012). In this section, we will discuss the experimental results of both systems on their respective tasks.

## 5.1 Evaluating Timex Normalisation

Timex normalisation, which has been explained in depth in Section 3.1, consisted in providing a standardised value from a given time expression. Hereinafter we present the results of our TimeNorm normaliser for Spanish, as well as some comments on the errors made by our system and other considerations.

### 5.1.1 Results for Normalisation

As we mentioned in Section 3.5, we needed to take into account the TempEval-3 Spanish training corpus during the building phase in order to know how to improve our grammar once the first version was finished. This was also done by Bethard (2013b), who used the TimeBank and the AQUAINT corpora to develop their TimeNorm grammar. We present in Table 9 the results declared in Bethard (2013b) and in Table 10 our own, separated in training and testing corpora.

| Corpora | Timexes | TIMEN | HeidelTime | TimeNorm |
|---|---|---|---|---|
| AQUAINT | 652 | 69.5 | 74.7 | **76.5** |
| TimeBank | 1,426 | 67.7 | 80.9 | **84.9** |
| TIMEN | 214 | **67.8** | 49.1 | 56.5 |
| TempEval-3 | 158 | 74.1 | 78.5 | **81.6** |

Table 9: Results for the original English grammar in Bethard (2013b).

| Grammar | Corpora | Timexes | Accuracy |
|---|---|---|---|
| English | AQUAINT | 652 | 76.53 |
| English | TimeBank | 1,426 | 80.43 |
| Spanish | TimeBank-ES Train | 1,269 | 83.53 |
| English | TempEval-3 Eval | 158 | 81.65 |
| Spanish | TimeBank-ES Eval | 234 | 80.77 |

Table 10: Results for English and Spanish normalisation.

The results in Table 9 show the good performance of the original TimeNorm system published in Bethard (2013b). It beats the other considered systems except for TIMEN on the TIMEN corpus, which is explained by Bethard mainly as format errors. However, when we evaluated the original TimeNorm English SCFG without any change on the grammar and with the same corpus and number of timexes (Table 10) we got an accuracy of 80.43, more than 4 points lower than the declared one and slightly lower than HeidelTime's result according to Bethard (2013b). In any case, the evaluation scores on the TempEval test set are the same, and so are the results on the AQUAINT corpus.

According to the scores reported by us in Table 10, the performance of our Spanish grammar is not far from the original one developed for English (less than 1 point in F1 score). Nevertheless, we must note the difference in the number of timexes in the corpora: the Spanish training corpus contains 89% of the number of timexes in the English training corpus, but the number of English timexes is 68% of the Spanish ones if we look at the evaluation corpora. Therefore, in Spanish we have a bit less to train and much more to test, which coincides with the gap in the number of files shown in Table 1. The absence of DCTs would not change this distance, which remains 88% and 69% respectively.

### 5.1.2 Discussion of the Normalisation Results

We have revised and classified the errors made by our system regarding the gold standard files. The examples of the next error types present the timex and the gold standard TIMEX3 value vs. our system's TIMEX3 value, if there exists:

- 11 errors due to a lack of rules in our grammar (e.g. *un momento dado de la historia*, *a given moment in history*: "PAST_REF"),
- 9 errors made by the heuristic rules (e.g. *noviembre*, *November*: "2000-11" vs. "1999-11"),
- 8 grammar errors (e.g. *del 20*, *of the 20*: "2000-03-20" vs. "1920"),
- 6 errors because of a lack of context (e.g. *0*: "P0Y", a period of zero years),
- 1 tagging mistake (e.g. *media hora*, *half an hour*: "PT0.5H", a period of half an hour, vs. "PT30M", a period of thirty minutes), and
- 10 too difficult expressions (e.g. *93/94*: "P1Y", a period of one year), expressions that need real world knowledge (e.g. *un largo periodo*, *a long period*: "PXM", a period of undefined months, vs. "PXX", an undefined period), or others.

Most of the errors that are due to a lack of rules or to grammar mistakes that could be easily solved, although some are too specific and infrequent to build rules for them. For example, the time expression *un momento dado de la historia* could be parsed if we added just one rule for it, but the expression appears only once in the whole corpus and there are no other similar structures to create a general rule for them. As we explained in Section

4.3, our aim was to build a linguistically coherent grammar rather than a task-oriented one. For this reason, we prefer to leave these cases unsolved and do not try to adapt our grammar to the normalisation task nor the test corpus.

Those cases where heuristic rules lead to a mistaken value do not have a direct solution, since the heuristics have been designed to match the most frequent sense of the timex class. Remember, for example, that past time spans are always preferred to future time spans, so *noviembre* is parsed to "1999-11" rather than the correct "2000-11". The TimeNorm alternatives to these heuristic rules require the creation of many sets of rules that would complicate the grammar, and still we could surely find ambiguities. On the contrary, these cases could be better normalised by taking context into account, as well as those errors where more context was needed. This approach, however, does not fit in the TimeNorm SCFG procedures, whereas new TimeNorm systems (Laparra et al., 2018; Xu et al., 2019) probably deal better with this problem.

Tagging mistakes can be solved by modifying the rules that transform TimeNorm representations in TIMEX3 value tags, creating new unit configurations or, like in the case of *media hora*, just changing in a grammar rule the way the time expression is considered. In this case, we could just indicate that "media hora" leads to "0.5 HOURS", instead of "30 MINUTES".

The rest of the mistakes are too infrequent to adapt to our grammar or impossible to solve given the architecture of the TimeNorm system.

Moreover, there are cases where the gold standard tagging seems incoherent. For example, in the next sentences[20] we have two very similar expressions, *hace tiempo* (*some time ago*) and *hace poco* (*a short time ago*), that receive very different TIMEX3 value tags: "PAST_REF" (a reference to the past) and "PXM" (a period of undefined months), respectively.

1. El mercado británico es dinámico y ha sido nuestra prioridad desde *hace tiempo*.
   (The British market is dynamic and has been our priority since *some time ago*.)

2. Tras estas cifras hay un enorme sacrificio que hasta *hace poco* era ignorado por el establishment español y catalán.
   (Behind these numbers there is huge sacrifice that was ignored until *a short time ago* by the Spanish and Catalan establishment.)

At first sight, both timexes could be annotated as "PAST_REF" or "PXX" (totally undefined periods). In the second sentence, we might know that the event has been some months before, so we could change "PXX" by "PXM". However, there is no clear reason to assign them so different tags. Observe that they have prepositions (*desde*, *since*; *hasta*, *until*) before that would transform the temporal sense of the expressions if we considered them (like in TimeNorm), but in TimeBank they are not even SIGNAL elements, as could be expected in TimeML (Pustejovsky et al., 2005). Indeed, TimeNorm usually provides a

[20]The translations of these timexes and their context in the sentences have been adapted to show the problem in Spanish.

more specific representation and can sometimes handle more complex linguistic situations than TimeML, even if the concrete grammars need several improvements for that. The SCATE annotation scheme (Bethard and Parker, 2016), which is related to these early TimeNorm representations, does improve the interpretation of timexes.

After analysing all the training and testing issues, we can see that the TimeNorm SCFG approach has some inherent disadvantages. As we saw when we described field values, the decision to consider all of these elements as assigned to an anchor instead of abstractly, like periods, generates normalisation errors according to TimeBank, but also regarding their true meaning. For example, *una noche* (*a night*) is clearly not anchored to any date, but the TimeNorm system forces the anchoring.

A similar thing happens with heuristic rules, which turn all the affected ambiguities to the most frequent case in the original work, but are the easiest way to deal with ambiguities given the system architecture. Having context into account would be a good alternative if we could, and it would also allow us to consider different anchors, since having only the DCT strongly constrains normalisations. Nevertheless, rule-based systems have shown to be better than data-driven ones, at least according to TempEval-3 results. Newer approaches, like the last TimeNorm systems (Laparra et al., 2018; Xu et al., 2019) should be compared to check whether context modelling obtains currently better results than these rule-based systems.

With regard to this context problem, we must recall that a grammar cannot distinguish an expression by its sense, but only by its form. We have, for example, a case in the Spanish training corpus that receives a wrong normalisation although it is strictly the correct one. It is the word *bimensuales* (*twice a month*), which, according to the corpus, takes the meaning of *bimestrales* (*once every two months*). In our grammar, *bimensuales* is parsed to the correct meaning, but here we need the common meaning. Therefore, the normative approach, that we took because it was the best to build a grammar as simple and compact as possible, is unable to model real language. Again, more recent systems, like the new TimeNorm (Laparra et al., 2018; Xu et al., 2019), might perform better at this task.

## 5.2 Evaluating Timex Detection and Normalisation

The TempEval-3 Task A for timex detection and normalisation, as explained in Section 3.1, consists of a timex detecting part and a second phase to normalise all the timexes that have been found at least partially. We present below the scores achieved by our best detection models and our results compared to those in UzZaman et al. (2012). Then, we will discuss the errors made by our system and comment on other related issues.

### 5.2.1 Results for Detection and Normalisation

We trained different embeddings to detect timexes using the Flair framework for the English and Spanish tasks. The best scores and the mean of five runs of the three best detection models are shown in Table 11, separated by language. Note that these scores are computed using the F1 micro score usually used in NER (Tjong Kim Sang, 2002; Tjong Kim Sang

and De Meulder, 2003), in which they take into account both the span and the type of the expression.

| Model | Best score | Mean |
|---|---|---|
| FlairEmbeddings(news) | **87.30** | **86.06** |
| WordEmbeddings(en), FlairEmbeddings(en) | 86.82 | 85.52 |
| WordEmbeddings(en), FlairEmbeddings(news) | 87.10 | 86.03 |
| WordEmbeddings(es), FlairEmbeddings(es) | **92.14** | **90.92** |
| WordEmbeddings(es), FlairEmbeddings(oscar) | 91.58 | 90.19 |
| PooledFlairEmbeddings(oscar) | 90.99 | 90.15 |

Table 11: Results for the best Flair detection models for English and Spanish.

These models have been built using three different types of embeddings, alone or stacked: Flair embeddings pre-trained on a news English corpus ("news") (Akbik et al., 2018), the Wikipedia for English ("en") and Spanish ("es") (Akbik et al., 2018) or the OSCAR Spanish corpus ("oscar") (Ortiz Suárez et al., 2019); FastText word embeddings for both languages (Bojanowski et al., 2017); and pooled Flair embeddings (Akbik et al., 2019b) pre-trained on the OSCAR Spanish corpus.

It can be observed that the best models for English and Spanish do not share the configuration, but the English combination of FastText and Flair is not far from the best English model regarding the mean. In fact, Akbik et al. (2018) recommend the stack of word embeddings and Flair embeddings as the best model. Moreover, it must be noted the gap between languages, reaching almost five points if we compare the best of each. This may suggest that the Spanish corpus is easier for timex detection. The matter of the number of timexes indicated before also applies here.

We evaluated our system on the official TempEval-3 metrics described in Section 3.5, which are Strict F1, Relaxed ("Rel") F1, precision ("P") and recall ("R"), and Value F1. Tables 12 and 13 present the best models that participated in the English and Spanish tasks for timex detection and normalisation, being TIPSem the reference system. The results for HeidelTime were extracted from Strötgen and Gertz (2015)[21].

| Model | Strict F1 | Rel F1 | Rel P | Rel R | Value F1 |
|---|---|---|---|---|---|
| TIPSem | 81.63 | 84.90 | **97.20** | 75.36 | 65.31 |
| HeidelTime | 81.78 | 90.71 | 93.13 | 88.41 | 78.07 |
| NavyTime | 79.57 | 90.32 | 89.36 | **91.30** | 70.97 |
| SUTime | 79.57 | 90.32 | 89.36 | **91.30** | 67.38 |
| ClearTK | 82.71 | 90.23 | 93.75 | 86.96 | 64.66 |
| Flair-TimeNorm | **91.39** | **93.63** | 96.90 | 90.58 | **78.65** |

Table 12: Results from English TempEval-3 Task A and our English system.

[21] `https://github.com/HeidelTime/heideltime/wiki/Evaluation-Results`.

| Model | Strict F1 | Rel F1 | Rel P | Rel R | Value F1 |
|---|---|---|---|---|---|
| TIPSem | 82.6 | 87.4 | 93.7 | 81.9 | 82.0 |
| HeidelTime | 85.87 | 90.13 | **96.02** | 84.92 | **87.47** |
| Flair-TimeNorm | **93.23** | **95.83** | 94.36 | **97.35** | 74.48 |

Table 13: Results from Spanish TempEval-3 Task A and our Spanish system.

We can see that, in TempEval-3, HeidelTime won the detection strict and relaxed F1-scores in Spanish and the normalisation in both languages, whereas in English ClearTK got the best strict F1 and NavyTime and SUTime the best relaxed F1. These results favour the idea that data-driven systems, like ClearTK, are better for detection, whereas rule-based systems, like HeidelTime, are the most suitable for normalisation.

In comparison with our systems, we achieve better results both in strict and relaxed F1 in the two languages, overcoming the TempEval-3 scores by far in strict (around 10 points in English and 8 in Spanish). We also get the best score in English normalisation (only by 1 point), but in Spanish we stay far even from the reference score (8 points to TIPSem, 13 to HeidelTime).

Our normalisation results reflect that, according to the TempEval-3 metrics, we have correctly normalised around 84% and 78% of the English and Spanish relaxed matches respectively, which more or less agree with the only-normalisation results in Table 10[22]. To obtain their results, for example, HeidelTime would have correctly normalised around 86% and 97% respectively; this disagrees with the only-normalisation evaluation done by Bethard (2013b) to compare English TimeNorm to English HeidelTime, in which this last one achieved 78.5 accuracy against TimeNorm's 81.6.

### 5.2.2 Discussion of the Detection Results

The errors made by our two detection systems are caused by similar reasons, and the gap between strict and relaxed matches is not large. Because of this, we will only comment on the Spanish errors. The mistakes on strict match can be distributed as it follows:

- False negatives:
  - 12 timexes that have not been detected (e.g. *su mejor año deportivo*, *his best sporting year*), and
  - 7 timexes with incorrect span (e.g. *temporada 93/94*, *season 93/94*).
- False positives:
  - 3 timexes that did not exist in the corpus (e.g. *un gran año*, *a great year*).

[22]Remember that here we do not normalise DCTs.

Some errors come from the complexity and low frequency in the corpus of the timex structures. For example, our system has not identified any of the words in *su mejor año deportivo* as a time expression, even if it contains the common word *año*, and only detects this word in the timex *lo que va de año* (*so far this year*).

In other cases, the tagging of the expressions in the gold standard are ambiguous, like those with the word *futuro* (*future*), which can be either a timex or an event. Indeed, there are cases in which it is not clear whether an expression is a timex, an event or none of them. According to the corpus, the expression in italics in the first of the next three sentences is a timex, the one in the second is not and the one in the third is an event. If we compare these so similar expressions, the only difference is that in the first the year we talk about is supposedly real (as it is denied) and past, the second one is hypothetical and future, and the third one may not last the period of a year[23]. Even if we conceive these differences as reasons for annotating in the way the corpus does, it is obviously difficult for a detector to see it.

1. Agassi, que no ha tenido *su mejor año deportivo* ni en el capítulo personal...
   (Agassi, who did not have *his best sporting year* neither in his private life...)

2. Agassi insistió que puede ser mejor jugador para volver a tener *un gran año*... [*sic*]
   (Agassi repeated that he can be the best player to have a great year again...)

3. ... debido a que el *año* escolar acaba de terminar.
   (... because the academic *year* has just ended.)

We also find some time expressions that are not explicit enough to be correctly detected. This is the case of *12* in the next first sentence, where the detector has not learned to consider it as a timex related to the unit after *ó* [*sic*] (*or*) that belongs to the next detected timex, *14 horas* (14 hours)[24]. However, this concrete error could be due to the accent on *ó*, which is an inadvisable alternative to the preferred *o* (Real Academia Española, 2010) and appears only once in the test corpus and never in the train corpus. On the contrary, the *o* conjunction does exist in a similar context in the training corpus, like in the second sentence.

1. ... que obligan a los obreros a trabajar hasta *12* ó *14 horas diarias* por salarios de miseria...
   (... that force the labourers to work even *12* or *14 hours every day* for a pitiful wage...)

2. ... los estadounidenses solamente están *tres* o *cuatro días*...
   (... Americans are only for *three* or *four days*...)

[23]The fact that this "year" ends, and thus does something, seems not enough as every period has the capacity to end.

[24]The word *diarias* (*daily*) constitutes another timex by itself, so it is not part of the previous one.

In other cases, expressions that could apparently be a timex are separated in various timexes, probably because TimeML cannot address such complexity. For example, our system detects *la segunda quincena de marzo* (*the second fortnight of March*) as a single timex, but in the corpus it is divided in *la segunda quincena* and *marzo*. TimeNorm could probably represent the whole expression if we provided a configuration for *quincena*, whereas TimeML cannot.

There are also mistakes regarding type matches. The next sentence shows a different case of ambiguity, where the timex *hace años* is annotated as DURATION in the corpus but we could think, as our system does, that it as an undefined DATE, which is possible to tag in TimeML.

1. ... muchos de los centenarios censados murieron *hace años*, aunque su voto por correo pueda llegar ahora...
   (... many of the registered hundred-year-old residents died *years ago*, although their postal vote might arrive now...)

All in all, we can see that our detection system works well enough, as expected for such a neural sequence-labelling approach. On the other hand, our normalisers have a good performance in connection to the detector, although in Spanish we stay much lower than other systems. Still, a deeper examination of all the results remains to be done.

# 6 Conclusion

The main objective of this project was to develop a Spanish grammar that allowed us to normalise time expressions with precision. At the same time, we investigated the difficulties to build such a grammar for a language other than English and the differences in time normalisation across languages, in this case, English, Spanish and, less exhaustively, in Italian. We chose the TimeNorm system (Bethard, 2013b) because its modular architecture allowed us to combine it with any detector, and so we also experimented with neural-based sequence labelling approaches (Akbik et al., 2018) for language-independent timex detection. The results obtained show that our approach is robust across languages for both tasks.

Our strategy when we built the Spanish TimeNorm grammar was based in three principles: to respect the structure of the system and work as much as possible on the source grammar, to prefer a normative approach rather than a descriptive one, and to build a linguistically coherent grammar rather than a task-oriented one. This strategy allowed us to make our grammar simpler and more compact than, for example, the Italian one (Mirza and Minard, 2015), but we cannot forget that the system has been directly designed for English. Nevertheless, we demonstrated that such a grammar can be built for inflected languages like Spanish achieving good results in the two tasks, although for others, like French, would need some of the decisions taken for the Italian grammar, for example, on tokenisation. Some of the commented insights about normalisation, however, are independent of both the language and of the framework used. Furthermore, we believe that the principle not to generate ad-hoc rules according to a specific benchmark makes our grammar useful for tasks on data from another domain.

Regarding timex detection, we have proved that it can be performed as a sequence-labelling task and that our neural-based approach is valid independently of the language. Our results support the idea from Verhagen et al. (2010) and UzZaman et al. (2012) that data-driven systems are better for timex detection, while rule-based systems perform better at timex normalisation. Nonetheless, more recent systems like the last TimeNorm approaches (Laparra et al., 2018; Xu et al., 2019) remain to be tested.

Despite this good performance, we should note the most relevant limitations of our TimeNorm grammars and the systems itself. Grammars are generally too sensitive to unknown expressions (*quincena*, *fortnight*) and minimal spelling changes and misprints (*trenta* instead of *treinta*, *thirty*), impose too rigid morphosyntactic and orthographic structures to the input expressions (*43 minutos, 43 minutes* vs. *minutos 43, minutes 43*), favour the creator's knowledge over the common sense (*bimensuales, twice a month* vs. *bimestrales*, every two months), and their heuristic rules are not enough to deal with ambiguities (normalisation to "2001-08-29" instead of "2001-09-05"). These and other issues are summarised at the end of Section 4 from the perspective of the construction of our Spanish grammar.

Moreover, the TimeNorm SCFG system ignores context and therefore limits the anchor to the DCTs and impoverishes disambiguation, apart from other structural problems that have already been indicated. Laparra et al. (2018) and Xu et al. (2019) suggest a different

TimeNorm approach that could cover some of these deficiencies. Finally, we shold also consider the limitations of the TimeML markup language, which also affect our normalisation approach. Bethard and Parker (2016) propose a new annotation schema to solve these shortcomings.

Our intention for future work is to evaluate our Spanish TimeNorm grammar on extrinsic tasks as well as on out-of-domain data to test our normalisation capabilities. Furthermore, we could also test our system on the Clinical TempEval (Bethard et al., 2015, 2016) although it only evaluates timex detection, compare it to the new TimeNorm approaches or adapt it to work along with event and temporal relation extraction systems.

# References

Rodrigo Agerri and German Rigau. Projecting heterogeneous annotations for named entity recognition. In *IberLEF@SEPLN*, 2020.

Alan Akbik, Duncan Blythe, and Roland Vollgraf. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/C18-1139.

Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. FLAIR: An easy-to-use framework for state-of-the-art NLP. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59, Minneapolis, Minnesota, June 2019a. Association for Computational Linguistics. doi: 10.18653/v1/N19-4010. URL https://www.aclweb.org/anthology/N19-4010.

Alan Akbik, Tanja Bergmann, and Roland Vollgraf. Pooled contextualized embeddings for named entity recognition. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 724–728, Minneapolis, Minnesota, June 2019b. Association for Computational Linguistics. doi: 10.18653/v1/N19-1078. URL https://www.aclweb.org/anthology/N19-1078.

Steven Bethard. ClearTK-TimeML: A minimalist approach to TempEval 2013. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 10–14, Atlanta, Georgia, USA, June 2013a. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/S13-2002.

Steven Bethard. A synchronous context free grammar for time normalization. pages 821–826, October 2013b. URL https://www.aclweb.org/anthology/D13-1078.

Steven Bethard and Jonathan Parker. A semantically compositional annotation scheme for time normalization. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 3779–3786, Portorož, Slovenia, May 2016. European Language Resources Association (ELRA). URL https://www.aclweb.org/anthology/L16-1599.

Steven Bethard, Leon Derczynski, Guergana Savova, James Pustejovsky, and Marc Verhagen. Semeval-2015 task 6: Clinical tempeval. pages 806–814, 01 2015. doi: 10.18653/v1/S15-2136.

Steven Bethard, Guergana Savova, Wei-Te Chen, Leon Derczynski, James Pustejovsky, and Marc Verhagen. Semeval-2016 task 12: Clinical tempeval. pages 1052–1062, 01 2016. doi: 10.18653/v1/S16-1165.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017. doi: 10.1162/tacl_a_00051. URL `https://www.aclweb.org/anthology/Q17-1010`.

Tommaso Caselli, Valentina Bartalesi, Rachele Sprugnoli, Emanuele Pianta, and Irina Prodanof. Annotating events, temporal expressions and relations in italian: the it-timeml experience for the ita-timebank. pages 143–151, 01 2011.

Nathanael Chambers. NavyTime: Event and time ordering from raw text. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 73–77, Atlanta, Georgia, USA, June 2013. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/S13-2012`.

Angel Chang and Christopher Manning. Sutime: A library for recognizing and normalizing time expressions. 11 2012.

J.-C Chappelier and Martin Rajman. A generalized cyk algorithm for parsing stochastic cfg. *TAPD'98 Workshop*, 01 1998.

Naomi Daniel and Timothy Allison. Sub-event based multi-document summarization. 03 2004. doi: 10.3115/1119467.1119469.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL `https://www.aclweb.org/anthology/N19-1423`.

Lisa Ferro, Inderjeet Mani, Beth Sundheim, and George Wilson. Tides temporal annotation guidelines. version 1.0.2. 2001.

David Graff. The aquaint corpus of english news text. 01 2002.

Benjamin Heinzerling and Michael Strube. BPEmb: Tokenization-free pre-trained subword embeddings in 275 languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 2018. European Language Resources Association (ELRA). URL `https://www.aclweb.org/anthology/L18-1473`.

Daniel Jurafsky and James H. Martin. *Speech and Language Processing*. Prentice Hall, 3rd edition draft edition, 2019.

Morgan Kaufman. Proceedings of the 6th conference on message understanding (muc6), 1995.

Morgan Kaufman. Proceedings of the seventh message understanding conference (muc-7), 1998.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1030. URL `https://www.aclweb.org/anthology/N16-1030`.

Egoitz Laparra, Dongfang Xu, and Steven Bethard. From characters to time intervals: New paradigms for evaluation and neural parsing of time normalizations. *Transactions of the Association for Computational Linguistics*, 6:343–356, 2018. doi: 10.1162/tacl_a_00025. URL `https://www.aclweb.org/anthology/Q18-1025`.

Chen Lin, Dmitriy Dligach, Tim Miller, Steven Bethard, and Guergana Savova. Multilayered temporal modeling for the clinical domain. *Journal of the American Medical Informatics Association : JAMIA*, 23, 11 2015. doi: 10.1093/jamia/ocv113.

Hector Llorens, Estela Saquete, and Borja Navarro. Tipsem (english and spanish): Evaluating crfs and semantic roles in tempeval-2. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 284–291, 2010.

Hector Llorens, Leon Derczynski, Robert J Gaizauskas, and Estela Saquete. Timen: An open temporal expression normalisation resource. In *LREC*, pages 3044–3051. Citeseer, 2012.

Inderjeet Mani and George Wilson. Robust temporal processing of news. In *Proceedings of the ACL'2000 Conference*, Hong Kong, oct 2000. Association for Computational Linguistics. URL `http://www.cs.yale.edu/homes/dvm/daml/`.

Paramita Mirza and Anne-Lyse Minard. HLT-FBK: a complete temporal processing system for QA TempEval. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 801–805, Denver, Colorado, June 2015. Association for Computational Linguistics. doi: 10.18653/v1/S15-2135. URL `https://www.aclweb.org/anthology/S15-2135`.

Pedro Ortiz Suárez, Benoît Sagot, and Laurent Romary. Asynchronous pipelines for processing huge corpora on medium to low resource infrastructures. 07 2019. doi: 10.14618/IDS-PUB-9021.

Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162. URL `https://www.aclweb.org/anthology/D14-1162`.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1202. URL `https://www.aclweb.org/anthology/N18-1202`.

J. Pustejovsky, R. Ingria, R. Saurí, José M. Castaño, Jessica Littman, R. Gaizauskas, A. Setzer, G. Katz, and I. Mani. The specification language timeml. In *The Language of Time - A Reader*, 2005.

James Pustejovsky, Patrick Hanks, Roser Saurí, Andrew See, Rob Gaizauskas, Andrea Setzer, Dragomir Radev, Beth Sundheim, David Day, Lisa Ferro, and Marcia Lazo. The timebank corpus. *Proceedings of Corpus Linguistics*, 01 2003.

Real Academia Española. *Ortografía de la lengua española*. Espasa-Calpe, Madrid, 2010.

Estela Saquete, José González, Patricio Martinez-Barco, Rafael Muñoz, and Hector Llorens. Enhancing qa systems with complex temporal question processing capabilities. *J. Artif. Intell. Res. (JAIR)*, 35:775–811, 08 2009. doi: 10.1613/jair.2805.

Roser Saurí and Toni Badia. *Spanish TimeBank 1.0*. 01 2012. ISBN 1-58563-620-7.

Andrea Setzer and Rob Gaizauskas. Annotating events and temporal information in newswire texts. 01 2000.

Jannik Strötgen and Michael Gertz. Heideltime: High quality rule-based extraction and normalization of temporal expressions. pages 321–324, 08 2010.

Jannik Strötgen and Michael Gertz. A baseline temporal tagger for all languages. pages 541–547, 01 2015. doi: 10.18653/v1/D15-1063.

Jannik Strötgen, J. Zell, and M. Gertz. Heideltime: Tuning english and developing spanish resources for tempeval-3. *Proc. of the 7th International Workshop on Semantic Evaluation*, pages 15–19, 01 2013.

William F. IV Styler, Guergana Savova, Martha Palmer, James Pustejovsky, Tim O'Gorman, and Piet C. de Groen. Thyme annotation guidelines. 2, 2014.

E. F Tjong Kim Sang and F. De Meulder. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147, 2003.

Erik F. Tjong Kim Sang. Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL-2002*, pages 155–158. Taipei, Taiwan, 2002.

N. UzZaman, Hector Llorens, James F. Allen, Leon Derczynski, M. Verhagen, and J. Pustejovsky. Tempeval-3: Evaluating events, time expressions, and temporal relations. *ArXiv*, abs/1206.5333, 2012.

Marc Verhagen, Rob Gaizauskas, Frank Schilder, M. Hepple, Jessica Moszkowicz, and James Pustejovsky. The tempeval challenge: Identifying temporal relations in text. *Language Resources and Evaluation*, 43:161–179, 06 2009. doi: 10.1007/s10579-009-9086-z.

Marc Verhagen, Roser Saurí, Tommaso Caselli, and James Pustejovsky. Semeval-2010 task 13: Tempeval-2. *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 57–62, 08 2010.

Dongfang Xu, Egoitz Laparra, and Steven Bethard. Pre-trained contextualized character embeddings lead to major improvements in time normalization: a detailed analysis. In *Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics (*SEM 2019)*, pages 68–74, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/S19-1008. URL `https://www.aclweb.org/anthology/S19-1008`.